МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Комп’ютерних наук, фізики та математики

Кафедра комп'ютерних наук та програмної інженерії

ПРОЕКТУВАННЯ ТА РОЗРОБКА СЕРВІСУ "SMARTUNIVERSITY"

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «бакалавр»

Виконав: здобувач 4 курсу 441 групи
Спеціальності 121 Інженерія програмного забезпечення

Освітньо-професійної (наукової) програми Інженерія програмного забезпечення

Войченко Владислав Володимирович

Керівник кандидат педагогічних наук, доцент, Вінник Максим Олександрович

Рецензент Tech Lead Forex Tester Software Морозов В'ячеслав Євгенійович

Херсон – Івано-Франківськ – 2023

Зміст

ВСТУП

Актуальність роботи

Актуальність проекту і теми полягає у стрімкому розвитку систем розумних приміщень та інтернету речей. Все більше підприємств та закладів починають використовувати інтернет речей для оптимізації та прискорення своєї роботи. Технологія розумних речей дозволяє заощаджувати час персоналу на перевірку всіх датчиків власноруч та значно зменшує час виявлення надзвичайних ситуацій, таких як пожежі, задимлення тощо.

Інтернет речей представляє собою мережу зв'язних пристроїв та програмного забезпечення, яке контролює їх, що дозволяє легко керувати системою та мати всю інформацію в одному місці. З кожним роком різноманітність моделей та марок пристроїв для системи інтернету речей зростає, що збільшує можливості для створення більш складних та корисних систем, які можуть автоматизувати більш складні процеси.

На даний момент інтернет речей ще не отримав великої популярності через складність інтеграції та експлуатації для великої кількості підприємств та через недовіру до нової технології, в якій можливі недоліки у безпеці, що може призвести до витоку конфіденційної інформації. Але всі ці проблеми вирішуються завдяки перевірці часом та усвідомленням необхідності інтеграції системи та навчання людей експлуатувати такі системи.

Майже в усі сфери життя можна інтегрувати інтернет речей. Такими установами можуть бути розважальні заклади, такі як кінотеатри та стадіони, де система може підраховувати заповнення посадкових місць та використовувати датчики для сповіщення про пожежі або інші аварії. Це можуть бути також такі установи як університети та лікарні, де також можна відслідковувати за надзвичайними ситуаціями, відслідковувати та контролювати температуру приміщень, планувати заходи в залежності

від кількості місць для людей в приміщенні, знаходити проблемні приміщення, де певні значення можуть бути поза межами норми, наприклад занадто висока чи занадто низька температура в кімнаті для перебування в ній людей.

В компанії IoT Analytics, що спеціалізується на дослідженні інтернету речей прогнозують зріст кількості активних підключень пристроїв до 27.1 мільярди [1]. В це число входять пристрої розумного будинку, гаджети, промислове обладнання та автомобілі, які мають пристрої з використанням інтернету. Зростання кількості пристроїв інтернету речей стимулюватиме розвиток та впровадження нових технологій та створення більш дешевих варіантів забезпечення розумних пристроїв інтернет зв’язком.

За прогнозом дослідницького агентства Gartner комунальні підприємства завдяки розвитку високотехнологічних лічильників стануть активними користувачами технології інтернету речей [2]. Що в майбутньому дозволить інтегрувати ці лічильники в готову систему розумного університету, для відслідковування витрат на електроенергії, газу тощо.

Актуальність розробки власного додатку для університету замість використання готового рішення, в тому, що додаток буде створено виключно під потреби університету та не матиме зайвого функціоналу. Університет матиме можливість змінювати додаток під свої потреби в будь який час без посередників у вигляді інших компаній. Також не буде необхідності витрачати кошти на підписку або придбання додатку, що дозволить заощаджувати кошти. Через те, що додаток буде власністю університету, доля додатку буде залежати від університету, а не компанії виробника, яка може припинити своє існування або може завершити підтримку додатку. Також незалежність від інших компаній дозволить

швидко вносити зміни без витрат часу та ресурсів на комунікацію з компанією-виробником додатку.

Також при створенні свого власного додатку університет сам обирає технології та мови програмування, за допомогою яких буде створено додаток. Обрання популярної технології, або тої, яка широко розповсюджена серед персоналу та студентів університету дозволить легше шукати розробників, які зможуть вносити зміни та підтримувати систему.

Мета роботи

Мета кваліфікаційної роботи полягає у розробці проекту «Smart University» з електронного моніторингу та задля керування університетом.

Для реалізації мети кваліфікаційної роботи необхідно:

- Пошук та огляд схожих продуктів
- Аналіз сильних та слабких сторін схожих продуктів
- Огляд та аналіз попередньої версії системи
- Створення вимог для нової версії системи
- Вибір технологій розробки системи
- Опанування необхідного інструментарію технології, якщо це необхідно
- Поетапна розробка системи за створеним планом зі створенням необхідних схем
- Тестування системи
- Виправлення помилок при необхідності

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПОДІБНИХ ПРОЕКТІВ

1.1 Пошук схожих проектів та встановлення пріоритетів

Перед початком створення системи треба оглянути та проаналізувати схожі роботи, що допоможе створити вимоги до продукту, знайти корисні рішення, проаналізувати помилки і недоліки сервісів та не допустити їх в процесі створення додатку. Знайти схожі проекти можна за допомогою пошукових систем та нейронної мережі ChatGPT-3.5[3], далі в тексті - ChatGPT.

За допомогою пошукових систем було знайдено такі роботи: Mi Home[4], CitiMan[5], Azure IoT Central[6], ThingsBoard[7][8][35], SmartCampus Sejong University[9], розумний кампус від Sònia Gudayol Marquès[10].

За допомогою запиту до нейронній мережі ChatGPT було отримано такі рішення: "Smart Campus" від компанії IBM, "Smart School" від компанії Huawei, "Smart Campus" від компанії Cisco. Але дослідивши джерела було визначено, що деякі з них припинили своє існування, а деякі надають доступ лише за запитом з контактними даними. Тому не один з перелічених варіантів не був цінним з точки зору дослідження. Запит нейронній мережі було сформовано на основі потреб проекту і написано у розмовному стилі. Запит виглядав наступним чином: «Мені потрібно зробити аналіз сайтів (або систем) конкурентів в сфері інтернету речей і в особливості систем розумного університету, розумної школи, розумного кампусу, які б сайти ти порадив для аналізу?». Далі для розширення бази дослідження було створено запит на пошук систем інтернету речей, які не були пов’язані з напряму з системами розумного університету, але які використовували такі технології. Запит було сформовано з урахуванням попереднього питання та відповіді на нього.

Запит мав наступну форму: «Добре, а тепер порекомендуй ще декілька сайтів пов'язаних з інтернетом речей, та не пов'язаних напряму з системами розумних університетів, шкіл та кампусів». На цей запит було отримано перелік сайтів, які публікували статті по темі інтернету речей, тому запит потрібно було скорегувати. Запит коригування виглядав наступним чином:
«Я не так висловився, мені необхідні готові проекти, типу Microsoft Azure», на що було отримано перелік продуктів для створення інтернету речей, які не були пов’язані на пряму з розумним університетом, школою та кампусом.
Так як проекти мали схожий функціонал, через їх схожість функціоналу, та мінімальну різницю у деяких функціях було прийнято рішення проаналізувати лише Google IoT.
Судячи з отриманої інформації з обох джерел було зроблено висновок, що через новизну технології існує не так багато відкритих для дослідження систем розумних університетів.

Як ми можемо зазначити – пошук за допомогою ChatGPT надав більш релевантні до теми проекту роботи, але аналіз яких не можливий або не має цінності в реалізації проекту, нейтральні по тематиці системи з використанням інтернету речей не мають великої цінності при розробці додатку розумного університету, так як їх функціонал не представляє аналітичного інтересу до теми проекту і ретельний огляд всіх функцій не раціональний з точки зору витрати часу.

Отже ми маємо наступний план аналізу:

Пріоритетний проект з детальним оглядом функцій:

- Розумний кампус від Sònia Gudayol Marquès
- Система пошуку шляхів в університеті STEERPATH

Другорядні проекти з поверхневим оглядом та аналізом функцій:

- SmartCampus Sejong University
- Mi Home
- CitiMan
- ThingsBoard
- Azure IoT Central
- Google IoT

1.2 Аналіз схожих проектів

Розумний кампус від Sònia Gudayol Marquès – В попередній роботі[11] було розібрано, що датчики працюють за допомогою трьох різних апаратних платформ. Розбір апаратних платформ системи не має практичної цінності для розробки системи розумного університету через те, що університет вже обрав платформу для роботи з датчиками і має готовий сервер, який надає інформацію з датчиків по запиту. Варто розглянути лише реалізацію фронтенду та архітектуру самого додатку.

Серверна частина, яка відповідає за створення сторінок була реалізована за допомогою Java Server Pages. Даний вибір технології не є оптимальним в реалізації системи розумного університету, тому що додає зайву технологію, що ускладнює написання та підтримку додатку, додаючи нову мову програмування та технологію.

Веб інтерфейс має мінімалістичний інформативний вигляд, який має ціль продемонструвати роботу проекту та автор не ставив метою створити повноцінний функціональний сервіс, тому спиратись на цю реалізацію не є доцільним до цього проекту, але концепція відображення та структура має схожу зі структурою запланованого проекту, тому варто більш ретельно звернути на неї увагу та проаналізувати.

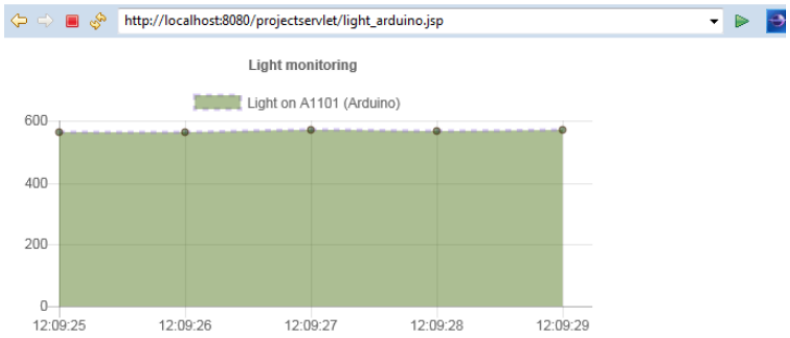

Рис. 1.1 Графік, який відображає рівень освітлення

На цьому графікі відображений рівень освітлення з датчика за часовими проміжками. Такий спосіб подання інформації можна використати в запланованому проекті, що дозволить аналізувати стан приміщення за певний відрізок часу та зручно виявляти проблеми.

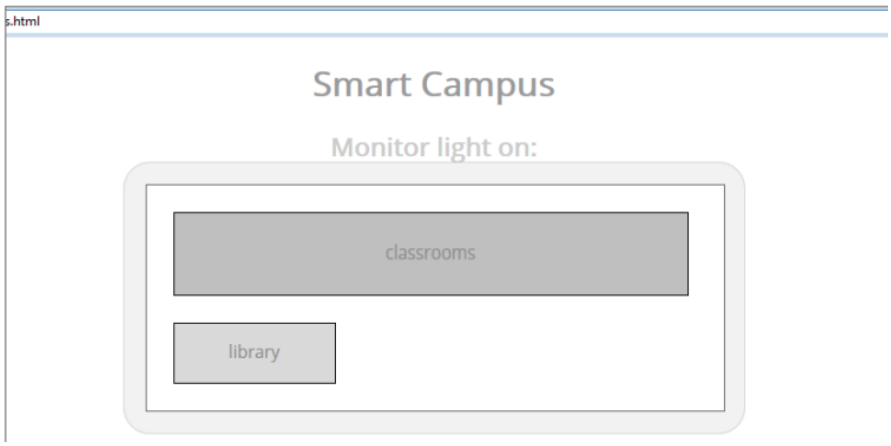

Рис. 1.2 Інтерфейс вибору приміщення

На даній сторінці користувача зустрічають дві групи приміщень – бібліотека та аудиторії. Обравши одну з категорій користувач або потрапляє на сторінку приміщення або переходить до наступного кроку – вибору більш дрібного залежного елементу, в даному випадку це буде вибір конкретної аудиторії. Схожа система вже реалізована на попередньому сайті і більш детально вона буде розглянута при аналізі старого сайту, фінальному створенні вимог та в описі реалізації проекту.

Система пошуку шляхів в університеті STEERPATH – Це система пошуку необхідних приміщень та орієнтації в будівлях

університету. Вона також дозволяє створювати статистику популярних місць в університеті, що дозволить оптимізувати або покращити саме ті місця, які використовуються більшістю, та визначити які зони та чому студенти не мають бажання відвідувати. Частина системи для інтерфейсу має вигляд мобільного додатку та веб сайту з мапами частини міста, де знаходиться кампус університету. Дозволяє обрати певні маркери на карті, перемкнути поверхи, тощо. Також додаток дозволяє знайти оптимальне місце для праці над проектами, видаючи студенту інформацію про кількість людей у приміщенні.

На офіційному сайті розміщена демонстраційна версія сервісу, яка дозволяє переглянути повний функціонал на заготовленому університеті.

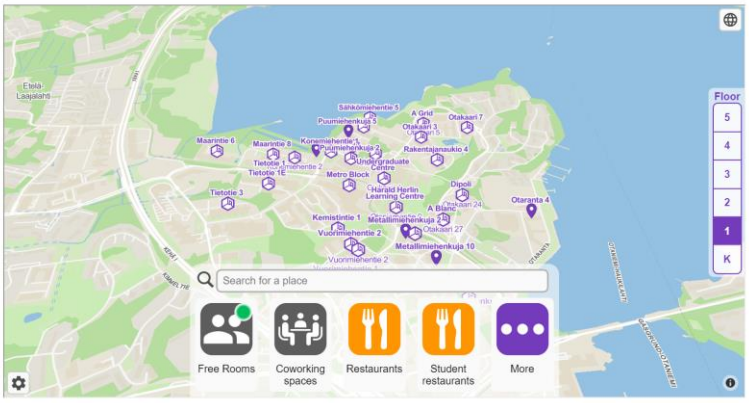

Рис. 1.3 Демонстраційне вікно додатку

Як можна помітити на карті стоять мітки з назвами будівель університету, задля швидкого пошуку необхідної частини кампусу. Також присутній пошук по назві, якщо студент не може знайти приблизне розташування приміщення, але почув про нього від персоналу університету або інших студентів.

Також можна обрати категорію приміщень і отримати список всіх приміщень даного типу, при натисканні на які відбувається приближення до обраного об’єкту. Також можна поділитись координатами приміщення за допомогою QR коду.

Всі приміщення відображені у вигляді 3D моделей, поверхи яких можна перемкнути в панелі справа.

Рис. 1.4 3D модель приміщення з позначками кімнат

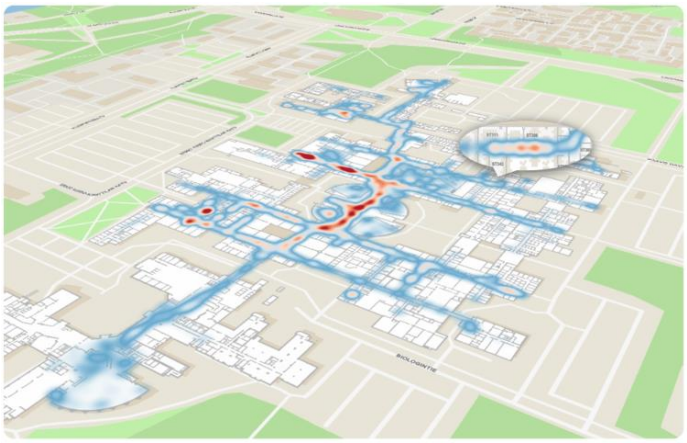

Рис. 1.5 Мапа популярності приміщень.

Мапа популярності дозволяє дізнатись які місця відвідуються найчастіше у зручному форматі, який не потребує детальної обробки та досліджень таблиць.

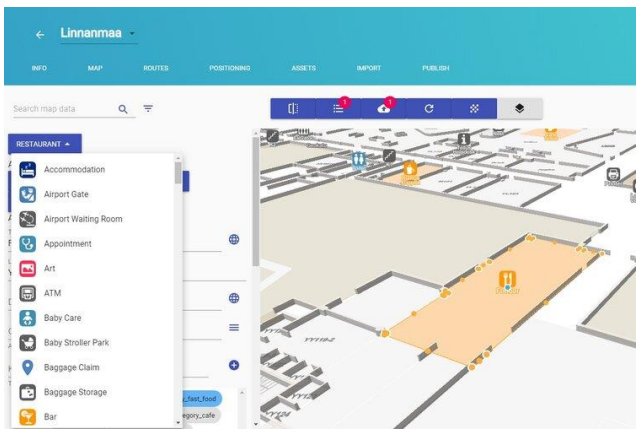

Рис. 1.6 Панель адміністратора. Режим редагування приміщень

Система дозволяє легко редагувати 3D модель приміщень, що дозволяє вносити зміни до мапи університету не звертаючись до висококваліфікованих професіоналів, які володіють специфічними навичками роботи з подібними системами.

SmartCampus Sejong University - Цей проект було оглянуто в попередній роботі [11], було визначено, що сервер створений за допомогою платформи NodeJS, яка буде використовуватись при розробці системи «Smart University». В системі присутня можливість отримувати дані про погоду в обраній географічній точці. Можливість реалізована за допомогою OpenWeather Api[12].
Також ця система може приєднуватись до датчиків інтернету речей за допомогою Mobius[13]. Що має велику схожість з запланованим проектом, але має деякі відмінності, у вигляді відсутності необхідних датчиків та має інший спосіб підключення до датчиків. В цілому проект не має великої дослідницької цінності, окрім самих основ створення серверу на NodeJs.

Mi Home – Цей проект як і попередні було оглянуто в минулій роботі [11]. Це комерційна система розумного приміщення, яка розроблялась с ціллю створення розумних будинків, але також може використовуватись для створення інших видів розумних приміщень. Керування системою реалізовано у вигляді мобільного додатку. Придбані датчики додаються до акаунту власника.

Користувач має змогу передивлятись стан датчиків, керувати ними самостійно та створювати сценарії роботи датчиків, наприклад циклічна активація через певний час, або при умові, що інформація з певного датчика рівна певному значенню.

Також можливо створювати скрипти, які запускаються користувачем, наприклад включення всіх пристроїв у певній кімнаті, включення та виключення світла у всьому приміщенні, тощо.

Деякими пристроями, які мають мікрофон, можливо керувати за допомогою голосу.

Також система дозволяє додавати користувачів в систему з різними рівнями доступу. Наприклад гість може отримати доступ лише до керуванням пристроями на кухні, а діти можуть мати доступ до всіх датчиків, але не матимуть права їх редагувати.

Підтримує лише сертифіковані виробником пристрої.

Станом час написання попередньої роботи система могла працювати з наступними видами датчиків [11]:

- Відеокамерами
- Розумними дзвінками
- Різними видами освітлювальних приборів
- Перемикачами живлення
- Датчиками температури, присутності, вікон та дверей, води, фото, руху
- Пристроями очищення повітря, кондиціонерами, обігрівачами, тощо
- Wi-fi маршрутизаторами, шлюзами, та посилювачами
- Динаміками
- Годинниками
- Кухонною електронікою
- Пристроями для прибирання
- Кормушками та поїлками, для рослин та тварин
- Зубними щітками, шторами
- Велосипедами, скутерами, відео реєстраторами

- Принтерами
- ІЧ-пультами

CitiMan – Проект був розглянутий у минулій роботі [11]. Він має вже готовий дешборд з готовим набором елементів інтерфейсу та скриптів, допускається модифікація під потреби адміністрації міста.

Основою функціонала є моніторинг за показниками у місті та різними видами попереджень, таких як попередження про затори, зайнятість всіх місць, заповнення всіх баків для сміття, тощо.

Дозволяє переглядати показники міста з датчиків з відокремленням по регіонам та вулицям міста. Нові райони та вулиці можуть бути додані самою адміністрацією міста.

Має сторінку з мапою міста, на якій відображаються мітки з датчиками, на які можна натиснути та отримати інформацію з датчика, кольорове підсвічення заторів та трафіку на дорогах, тощо.

Також користувач може створювати правила та обмеження для міста, наприклад вимкнення вуличного освітлення в певний час доби, увімкнення світла в певній ділянці міста лише при наявності людей, ввімкнення освітлення лише при паркуванні автомобіля, тощо.

Система може генерувати звіти про стан міста з вказаною періодичністю. Звіти включають в себе електроспоживання, зайнятість місць для паркування, статистику значень по датчикам, тощо.

Серед основних показників міста з датчиків є [11]:

- Погода
- Світло
- Середня гучність
- Відеокамери
- Кількість wifi мереж та користувачів

- Моніторинг паркінгу
- Розумні сміттєві баки
- Розуме вуличне освітлення
- Використання електроенергії
- Трафік

Інтерфейс додатку виглядає наступним чином:

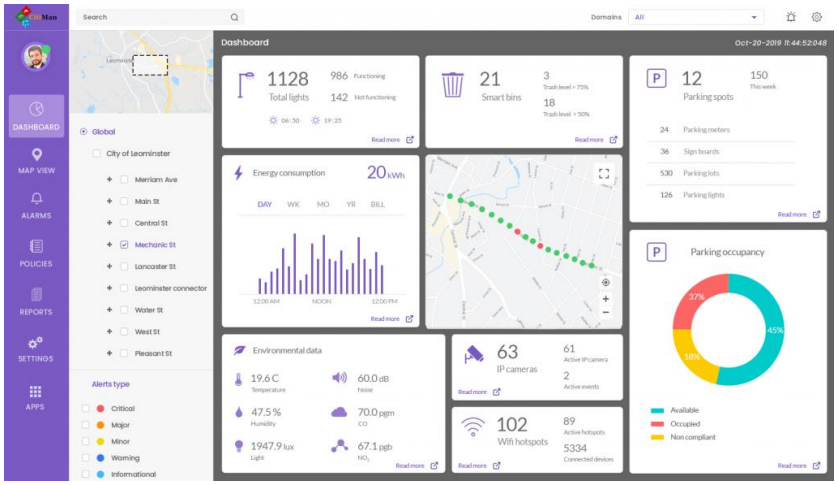

Рис. 1.7 Веб-інтерфейс CitiMan

Як можна побачити, то він складається з трьох частин – навігаційного меню зліва, дешборду з основною інформацією справа та з панеллю інструментів та налаштувань по середині.

ThingsBoard – Цю open-source платформу було оглянуто та досліджено в попередній роботі [11]. Інтерфейс проекту побудований у вигляді дешборду, де користувач може обрати готовий пресет з датчиків та елементів інтерфейсу, що дозволить швидко розпочати роботу, або протестувати систему перед створенням свого проекту. Також користувач може почати пустий проект, та збудувати інтерфейс під власні потреби за допомогою вже готових елементів, або зі створення власних за допомогою програмного коду.

Сервіс дозволяє додавати будь які датчики, які підтримують обмін інформацією через CoAP, MQTT та HTTP.

Кожен датчик можна відмітити для зручної роботи при наявності великої кількості пристроїв.

Правила для датчиків створюються на мові програмування у вигляді блок схем, а створення нових та редагування старих відбувається за допомогою JavaScript.

Сервіс є платним і має декілька видів передплат, але пропонує демонстраційну безкоштовну версію на місяць.

Інтерфейс платформи виглядає наступним чинином:

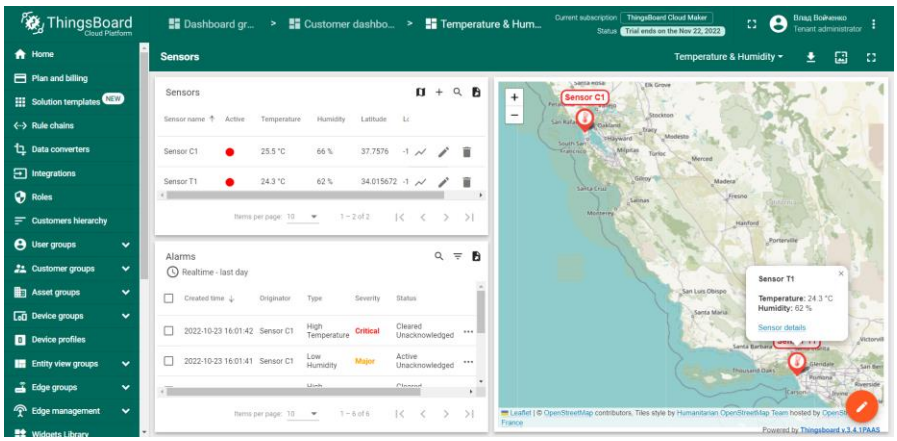

Рис. 1.8 Веб-інтерфейс ThingsBoard

Як можна помітити, інтерфейси з різних платформ мають схожий вигляд та структуру. Схожий дизайн і структуру можна зробити й в проекті розумного університету, для сторінки керування сайтом.

Сторінка створення правил та скриптів виглядає наступним чином:

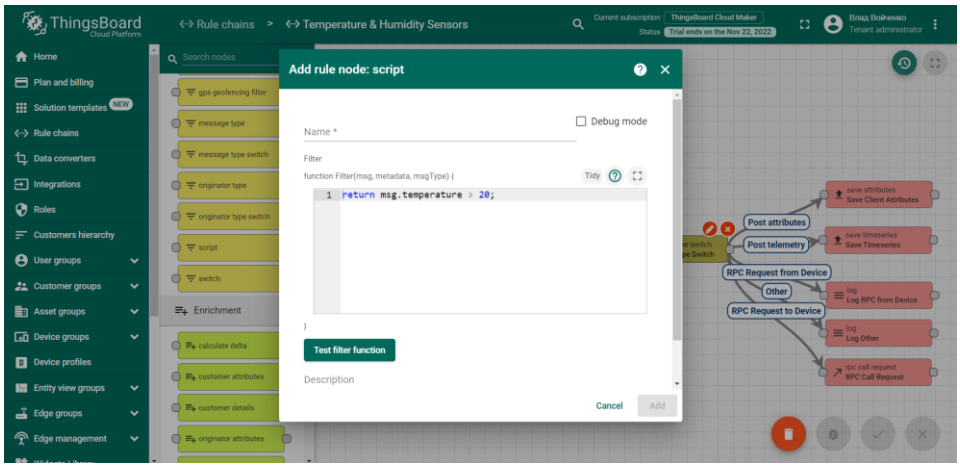

Рис. 1.9 Вікно написання скрипта

Вікно має стандартну для подібних додатків структуру, вікно створення блок схем виглядає аналогічно до більшості з існуючих на ринці програм, з побудови блок схем. Але додавання такого функціоналу не є необхідним для проекту «Smart University», тому що він не націлений на широку аудиторію та комерційне використання.

Azure IoT Central – Платформа для масової обробки інформації з датчиків [11] та створення програм для керування роботою датчиків та для створення систем з датчиками. Має більший бюджет, внаслідок чого більше можливостей і функціоналу. Додатково дозволяє відслідковувати вакансії в певній сфері, відображати повідомлення з датчиків, які надходять у форматі JSON, та більше не названих можливостей.

Платформа дозволяє розробляти проекти на багатьох мовах програмування, що збільшує кількість потенційних розробників та користувачів, серед мов програмування є наступні[11]:

- .NET
- Java
- Node.js
- Python
- PHP

- Go

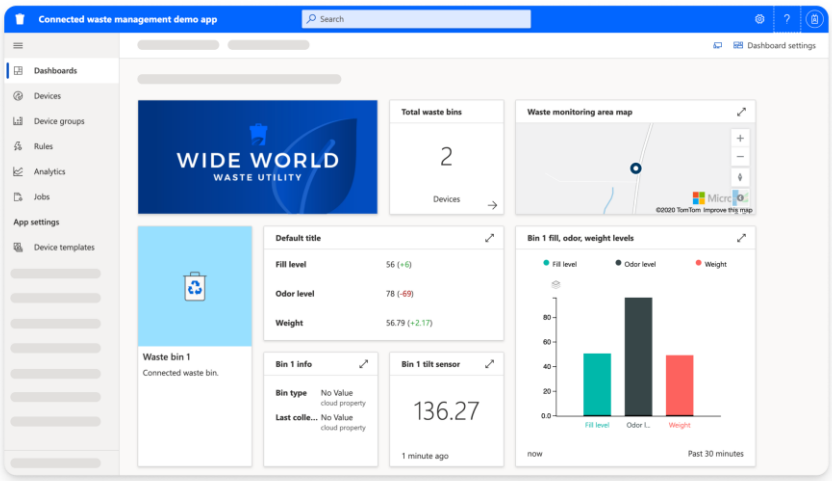

Рис. 1.10 Інтерфейс платформи.

Як можна помітити інтерфейс додатку створений також у вигляді дешборду з подібним до інших розташуванням елементів, що вказує на те, що оформлення користувацького інтерфейсу в такому вигляді є оптимальним з точки зору більшості компаній.

Google IoT – Комерційна платформа для великих компаній, яка дозволяє організовувати роботу систем інтернету речей та має більш технічний користувацький інтерфейс, що робить користування платформою незручною для звичайного користувача, який не має навичок в сфері створення програмного забезпечення та інтернету речей. Дозволяє додавати датчики, отримувати інформацію про їх стан, налаштовувати та відслідковувати інформацію з датчиків.

Припинення роботи системи заплановано на 16 серпня 2023 року.

Ця система не має інформаційної цінності для створення архітектури проекту, але може надати інформацію по створенню та організації фронтенд частини проекту.

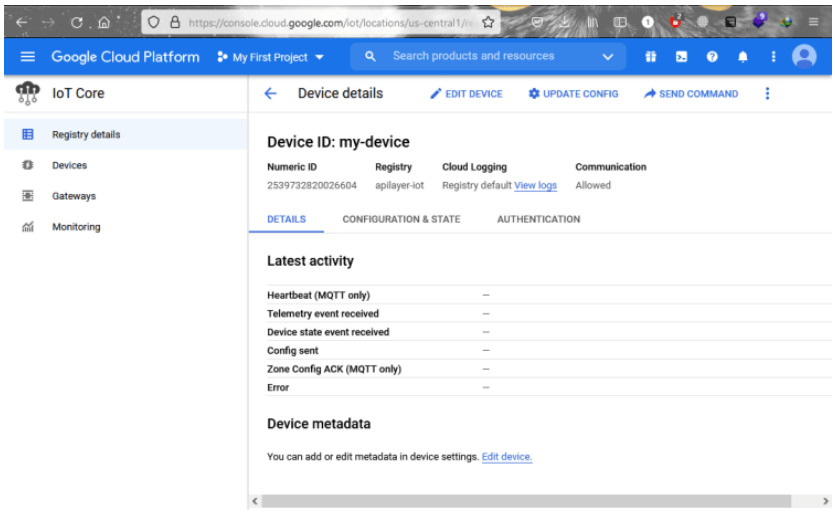

Рис. 1.11 Інтерфейс Google IoT Cloud Platform

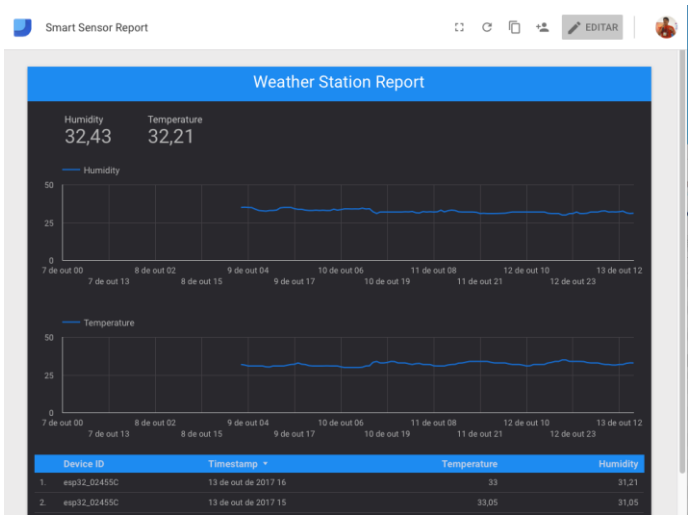

Рис. 1.12 Приклад реалізації сторінки станції моніторингу погоди

1.3 SWOT аналіз схожих робіт

- Розумний кампус від Sònia Gudayol Marquès
- Система пошуку шляхів в університеті STEERPATH

Другорядні проекти з поверхневим оглядом та аналізом функцій:

- SmartCampus Sejong University
- Mi Home
- CitiMan
- ThingsBoard
- Azure IoT Central

- Google IoT

На основі проаналізованих сервісів можна зробити опис функціоналу, які сервіси мають, порівняти їх та зробити висновки щодо доречності тих чи інших функцій проекті «Smart University».

Розумний кампус від Sònia Gudayol Marquès

- Датчики освітлення
- Датчики температури
- Датчики присутності
- Силові датчики
- Магнітні датчики
- Історія значень з датчиків
- Схематичне зображення університету
- Вибір приміщень по категоріям

Серед всіх функцій найбільш придатними до використання в проекті «Smart University» є датчики температури, так як це дозволить автоматизувати процес збору інформації про стан приміщень, який проходить на даний момент у ручному режимі з перенесенням термометра від аудиторії до аудиторії.

Датчики освітлення, присутності, магнітні датчики та силові не мають необхідності і збільшують вартість проекту.

Схематичне зображення університету можна замінити на більш ілюстративні схеми будівель та поверхів для більш зручного користування.

Історія значень з датчиків є важливою частиною проекту, яка дозволить аналізувати стан приміщень за великий проміжок часу, виявляти тенденції, постійні проблеми та вирішувати їх.

Система пошуку шляхів в університеті STEERPATH

- Карта університету
- Мітки приміщень
- 3D моделі приміщень
- Опис приміщень
- Популярність місць
- Різні рівні доступу
- Панель адміністратора

Система має доволі детальну схему приміщень університету, дозволяє редагувати моделі з панелі адміністратора, дозволяє імпортувати моделі з програм для моделювання.

При початковому обговоренні проекту було вирішено, що будуть використовуватись SVG зображення приміщень університету, тому деякі функції, які можна реалізувати в двомірному просторі можна додати в проект розумного університету.

При натисненні на приміщення з’являється більш детальна інформація про них, це можна використати в проекті, щоб допомогти студентам та партнерам орієнтуватись в приміщеннях.

Панель адміністратора, яка дозволяє зручно редагувати інформацію дозволить зменшити навантаження на програмістів, які можуть бути зайнятими більш важливими проектами.

Створення різних рівнів доступу дозволить розподілити навантаження по редагуванню та створенню бази даних на більшу кількість людей, які відповідальні за конкретні поверхи, факультети та будівлі.

SmartCampus Sejong University

- Моніторинг погоди

Функцію моніторингу погоди можна розглянути як потенційне розширення системи при завершенні реалізації основного функціоналу. Функція не пріоритетна, так як існує багато сайтів з погодою в інтернеті до яких люди вже звикли.

Mi Home

- Розумна розетка
- Розумний перемикач
- Датчики температури
- Датчики вуглекислого газу
- Різні рівні доступу
- Датчики якості повітря
- Розумні лампочки, світильники
- Відеокамери
- Даниміки
- Годинники
- Кухонна електрика
- Пристрої для прибирання
- Кормушки та поїлки
- Принтерами
- ІЧ пультами

Такі пристрої як розумні розетки та перемикачі потенційно можна додати до системи, щоб створити систему дистанційного управління університетом і полегшити роботу персоналу, якому зараз необхідно проходити велику відстань для того щоб просто вимкнути пристрій або освітлення.

Датчики вуглекислого газу та якості повітря є пріоритетним функціоналом системи, вони необхідні для облегшення щоденної праці персоналу і покращення умов для студентів та працівників університету.

Розумні лампочки та світильники також можуть використовуватись для облегшення роботи персоналу, це дозволить вимкнути всі джерела освітлення не переміщаючись по університету фізично.

Динаміки, годинники, кухонна електрика, пристрої для прибирання, кормушки, принтери та ІЧ пульти не є приоритетними датчиками і можуть принести більше збитків, ніж буде отримано користі від них, тому ці пристрої потребують додаткового економічного аналізу.

Розумні відеокамери можуть бути поєднані з системою розумного університету, але великої необхідності на початку розвитку проекту це не має.

CitiMan

- Погода
- Розумне освітлення
- Датчики гучності
- Відеокамери
- Кількість користувачів у WiFi мережах
- Моніторинг паркінгу
- Розумні сміттєві баки
- Лічильники електроенегрії

Такі пристрої як датчики гучності та кількість користувачів WiFi точок можуть бути корисними при аналізі просторів університету, але не є пріоритетними при реалізації системи розумного університету.

Лічильники енергії можуть полегшити роботу персоналу та бухгалтерії, тому варто звернути увагу на ці пристрої.

ThingsBoard, Azure IoT Central, Google IoT

Універсальні платформи, які підтримують будь які датчики, які можуть обмінюватись через CoAP, MQTT та HTTP

Створенния блок-схем та скриптів

Ці системи є складними і реалізація більшості функціоналу не має практичного та економічного сенсу в реалізації системи розумного університету.

Висновки з аналізу проектів:

Деякі функції та види датчиків можна додати в систему розумного університету, але повністю копіювати проекти не варто, тому що всі проекти мають різне направлення і цілі і деякий функціонал не матиме сенсу конкретно для цього проекту.

1.4 Огляд та аналіз минулого сайту

Минулий сайт має готовий фронтенд, який містить наступні сторінки:

- Головна
- Інформація про факультети
- Сторінки корпусів
- Сторінки усіх поверхів

Всі сторінки окрім сторінок поверхів мають статичні сторінки, інформація яких не змінюється, тому є сенс розглянути лише сторінки поверхів з точки зору реалізації функціоналу розумного університету.

Першим елементом, який повинен бути інтерактивним є схематична карта поверху. Вона статична і відображає назву приміщення при наведенні на нього, кількість місць, факультет, тип приміщення та закріплену кафедру.

Рис. 1.13 Схема поверху з підказкою до аудиторії 507

Цей елемент має статичний вигляд і редагується програмістом. Його функціонал можна покращити додаванням посилань на розширені сторінки аудиторій з більшою кількістю інформації, яка буде братись з серверу та може бути відредагована не тільки програмістом. Спливаючі підказки можна генерувати використовуючи інформацію про аудиторію з бази даних.

Також можна додати функціонал розміщення вже створених в панелі адміністратора приміщень до комірок на схемі поверху, при відсутності прив’язаної аудиторії та при наявності прав адміністратора можна перенаправляти на сторінку прив’язки аудиторії до комірки. При відсутності ж таких прав можна динамічно прибирати посилання та спливаючі підказки з комірки, замінюючи їх на текст про відсутність інформації.

В деяких комірках є кнопка з переглядом фотографій аудиторії, але всі ці фотографії додаються так само статично програмістом. Функцію перегляду фотографій можна перемістити на сторінку аудиторії, де фотографії будуть отримуватись з бази даних.

Наступним блоком є інформація про поверх та дані з датчиків вуглекислого газу та термометра.

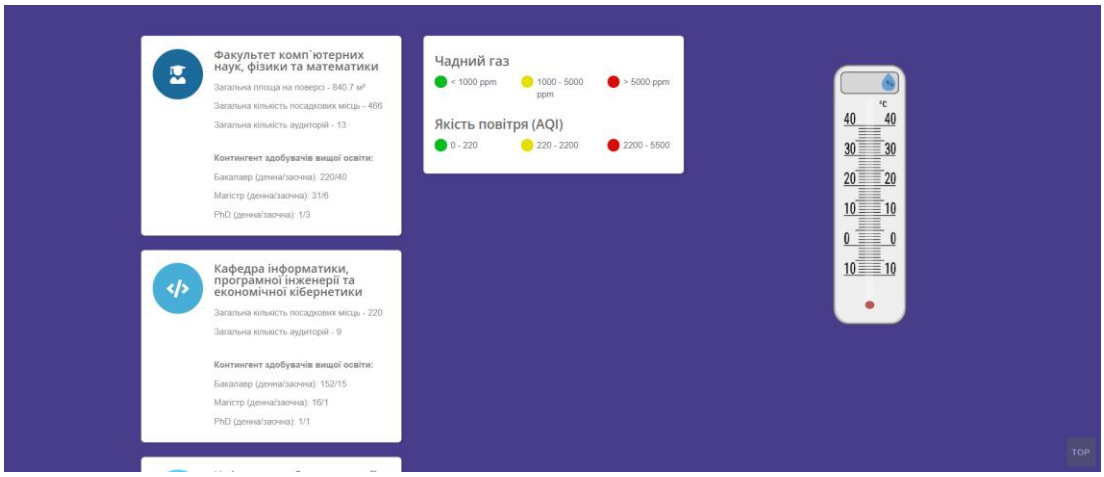

Рис. 1.14 Блок з інформацією про поверх

Блок інформації про факультет та кафедри можна генерувати на сервері враховуючи інформацію з бази даних та відправляти користувачу фінальний результат.

Блок з інформацією про чадний газ та якість повітря виглядає як кольоровий індикатор та опис кольорів. При первинному огляді сайту деякі датчики не працювали і кольоровий індикатор не з’являвся. Відсутність інформації про помилку отримання інформації може заплутати користувача, який намагатиметься зрозуміти принцип роботи непрацюючого датчика.

Блок термометра відображає температуру на шкалі термометра та вологість над шкалою. При наведенні на блок він збільшується. Сенсу зменшувати чи збільшувати датчик немає, так як по всім бокам від блоку є вільний простір, який блок займає, тому цю функцію можна прибрати і відображати завжди у збільшеній формі. При необхідності можна зробити розширення у вигляді popup.

Також можна змінити вигляд датчика і подавати інформацію в цифрах, а не у вигляді датчика, що буде більш зручно для сприйняття користувачем.

При аналізі коду датчиків було помічено, що у фронтенд частині скрипта, який завантажується користувачу, при запиті на сервер використовується конфіденційна інформація з логіном та паролем від інформації датчика. Її необхідно перенести на сервер, а сервер вже видаватиме підготовлену інформацію по API. Також код має не оптимальний вигляд, наприклад пошук елементів сайту відбувається не при ініціалізації коду, а в функціях, що змушує користувача витрачати більше ресурсів комп'ютера при роботі з сайтом, тому код потребує доопрацювання.

Далі знаходиться блок зі статистикою по поверху. В залежності від сторінки там відображається різна інформація: Доля типів приміщень, доля кафедр, кількість посадкових місць.

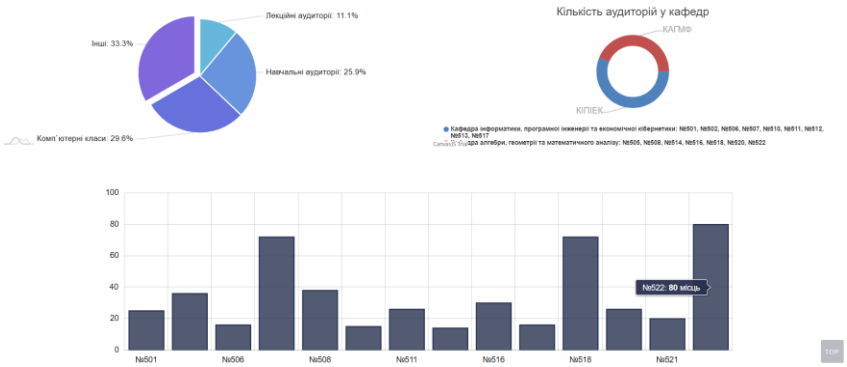

Рис. 1.15 Блок статистики

Ці графіки генеруються на основі інформації зі статичного json файлу. Цю інформацію можна динамічно генерувати на сервері з використанням інформації з бази даних та відправляти по API.

Також є сенс розглянути фронтенд частину на наявність проблем та пошук можливостей покращити код.

Розпочати можна з огляду меню.

Рис. 1.16 Меню сайту

Воно має декілька недоліків у вигляді відсутності простору над логотипом, та великим відступом знизу. Варто відцентрувати логотип для кращого сприйняття сайту.

При зміненні режиму меню з кольорової схеми в чорно-білу логотип на короткий чай зникає, що створює враження низької якості сайту у користувача.

Також позначка обраної сторінки завжди стоїть на головній. Це можна виправити додаванням динамічної генерації меню на сервері.

При перегляді сайту меню займає велику частину екрану, чим зменшує корисну площу екрану, що робить сприйняття інформації менш комфортним. Доречно було б плавно змінювати висоту меню, якщо користувач не на верху сторінки.

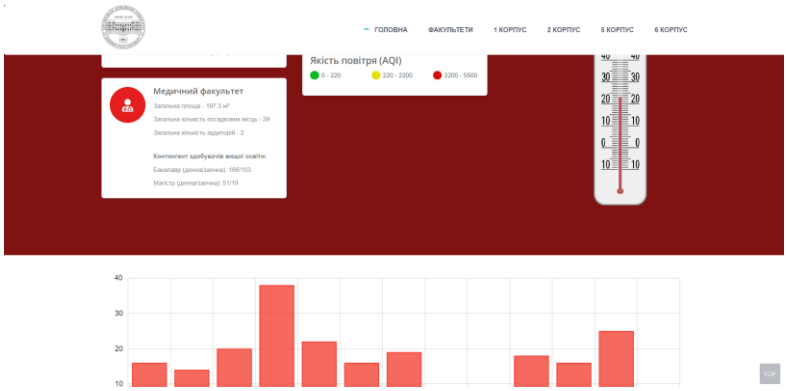

Рис. 1.17 Сторінка на моніторі ноутбука

Головна сторінка сайту має слайдер, який має такі недоліки як занадто малі кнопки керування слайдами, що змушує користувача витрачати багато часу на наведення мишкою. Також слайдер просто дублює той самий текст на кожен слайд, хоча доречно було б додати його просто поверх слайдеру. Також варто було б додати автогортання слайдів, тому що наявність слайдеру може бути неочевидною для користувача, а сам слайдер має лише декоративну роль

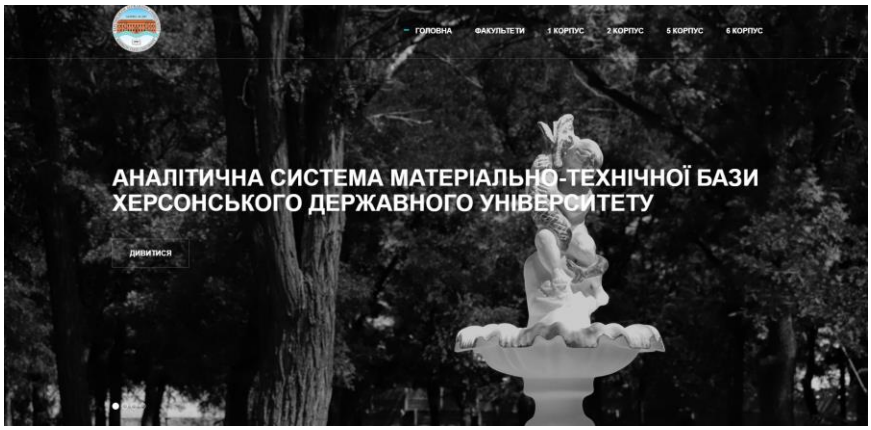

Рис. 1.18 Слайдер на головній сторінці сайту

Деякі посилання на сайті, наприклад такі як в футері сайту не мають плавної анімації при наведенні, що вибивається з дизайну сайту з плавними анімаціями на більшості об’єктів.

При подальшому огляді комп’ютерної версії сайту значних недоліків знайдено не було.

Тепер варто оглянути та проаналізувати мобільну версію сайту на наявність багів та недоробок.

В першу чергу звертається увага на меню, яке має аналогічну проблему з десктоп версією – воно займає забагато корисної площі екрану у користувача, що робить роботу з сайтом не комфортною

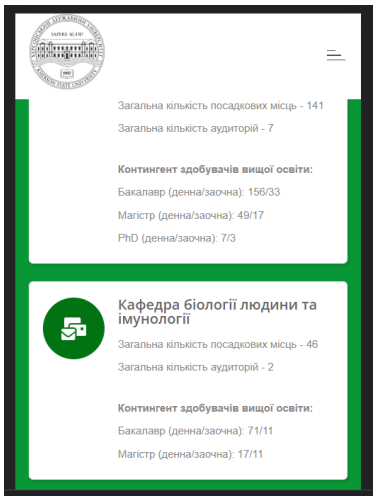

Рис. 1.19 Мобільна версія сайту

Розгорнута версія меню сайту має деякі недоліки з побудовою, такі як можливість гортати сторінку при відкритому меню, що робить користування сайтом менш інтуїтивним та може призвести до небажаного гортання сторінки користувачем, замалі пункти, що робить користування сайтом не комфортним в русі або на малому екрані та завеликий логотип, який займає дуже багато корисного місця на екрані.

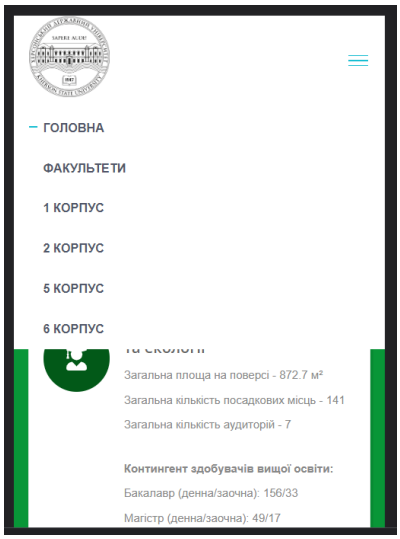

Рис. 1.20 Відкрите меню на мобільній версії сайту

Деякі елементи сайту такі як статистика знаходяться за межами екрану. Що зменшує можливості сайту при роботі на мобільному пристрої.

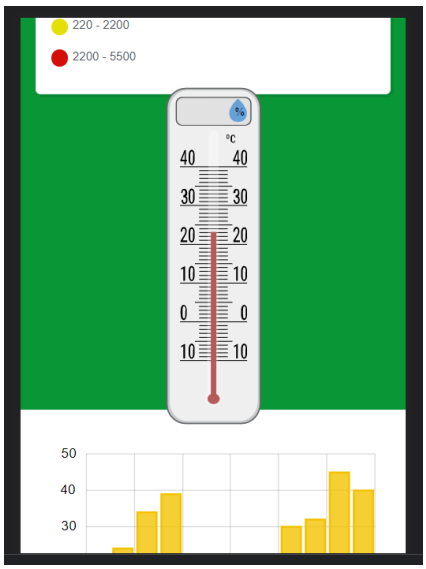

Рис. 1.21 Кінець сторінки, обрізана статистика

Блок термометра при натисканні збільшується на незначний відсоток. Об’єктам, які мають дії при наведенні на комп’ютерній версії сайту, краще заміняти ці дії або постійним ефектом від наведення, або його повним видаленням, або заміну на натискання, якщо це необхідно.

Сторінки вибору поверху мають незручне та неінтуїтивне розташування блоків, варто додати список з поверхами до гори сторінки, а зображення знизу, щоб користувач зміг зорієнтуватись до чого стосується зображення корпусу. Дії при натисканні на поверхи треба прибрати, так як зображення замале. Другий блок з вибором поверхів треба прибрати за непотрібністю.

Сторінка з інформацією про факультети містить таблицю, яка перетворюється в одну колонку на мобільній версії, яка має відступи з різною відстанню, хоча інформаційної цінності різні відступи не несуть, тому краще привести їх до однакового розміру. Футер та деяка інформація також зникає на мобільній версії.

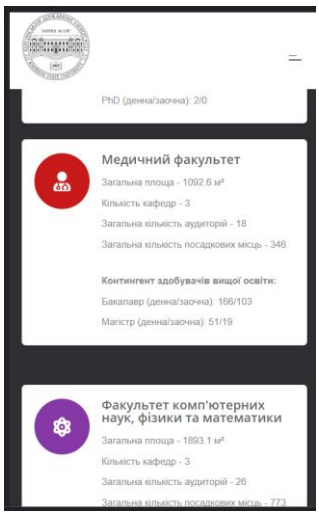

Рис. 1.22 Сторінка факультетів, різний розмір відступів

РОЗДІЛ 2
ТЕХНІЧНА СПЕЦИФІКАЦІЯ ПРОЕКТУ

2.1 Технології

В проекті будуть використані наступні технології: Для реалізації серверної частини буде використана платформа NodeJs[14], також будуть використані такі бібліотеки: ExpressJs[15] для роботи зі сторінками, Mongoose[16] для роботи з базами даних, Multer[17] для роботи з файлами, express-session для роботи з сесіями користувачів, cookie-parser[18] для роботи з кукі файлами, bcryptjs[19] для роботи з шифруванням, jsonwebtoken[20] для створення токенів користувачів. Та деякі інші малі бібліотеки, які будуть згадані в процесі створення проекту.

Для реалізації фронтенду буде використано такі технології: Фреймворк VueJs[21] для зручної роботи з фронтендом, ExpressJs, для генерації сторінок та деяких елементів фронтенду, bootstrap[22] для допомоги при верстці, Gulp[23] для швидкої та зручної верстки та деякі інші технології, які будуть згадані в процесі роботи або можуть бути знайдені у файлах проекту на GitHub.

База даних буде реалізована за допомогою MongoDB[24], також для дебагу та швидкого редагування буде використовуватись інтерфейс MongoDBCompass[24]. Для перевірки роботи API буде використовуватись утиліта для створення запитів Postman[25].

Код проекту буде розміщено на сторінці GitHub[26].

Для демонстрації проекту під час розробки буде використовуватись сервіс Ngrok[27].

Для збору інформації та допомозі у вивченню деяких бібліотек буде використовуватись ChatGPT-3.5[3].

2.2 Реєстр вимог

Реєстр вимог до системи було створено у попередній роботі [11], в даній главі його буде доповнено та додатково проаналізовано для створення більш чіткої картини проекту.

Список вимог в попередній роботі виглядав наступним чином [11]:

- Лише дві ролі - гість та користувач
- Адміністратор повинен мати можливість вийти з профілю
- Технологія - веб додаток, який отримує інформацію з сервера
- Інтерфейс повинен бути реалізованим у вигляді дешборду
- Має працювати з датчиками CO2
- Має працювати з датчиками температури
- Система має отримувати кількість підключень до мереж wifi
- Веб додаток повинен мати систему сповіщень користувача, при виході значень за встановлені межі
- Система повинна мати можливість керувати розумними розетками
- Всі датчики повинні бути згруповані по аудиторіям
- Аудиторії повинні бути згруповані по поверхам
- Поверхи повинні бути згруповані по факультетам
- Факультети повинні містити в собі сутності кафедр
- Кафедри повинні посилатись на пов'язані з ними аудиторії
- Аудиторії та поверхи повинні мати поле з номером
- Факультети та кафедри повинні мати поле з назвою
- Аудиторії повинні містити такі поля як:
    - Короткий опис
    - Посилання на 3д модель
    - Посилання на фотографії
    - Історію значень

    - Кількість посадкових місць
    - Тип аудиторії (лекційна, комп’ютерна, лабораторія)
    - Ім'я та прізвище закріпленого лаборанта
    - Прив'язка до кафедр
- Сутності wifi точок повинні бути окремими від факультетів
- Сутності wifi точок повинні мати ім'я та кількість користувачів
- Сутність користувача повинна мати лише логін та пароль
- Гість повинен мати доступ лише до сторінки входу в профіль
- Лише старші сутності повинні створювати молодші (Факультети - кафедри та поверхи, поверхи - аудиторії)
- Аудиторії повинні мати метод додавання кафедри, з якою вони пов'язані

Через деякі зміни та додаткове обговорення було прийнято рішення скоригувати такі пункти як:

- Лише дві ролі - гість та користувач
- Аудиторії повинні містити такі поля як:
    - Посилання на 3д модель
- Гість повинен мати доступ лише до сторінки входу в профіль
- Аудиторії повинні мати метод додавання кафедри, з якою вони пов'язані

Перший пункт було замінено іншими ролями – Адміністратор, Редактор факультету, Лаборант, Гість.

Ідея про реалізацію аудиторій у вигляді 3D була замінена на зображення у форматі SVG.

Гість також повинен мати доступ до перегляду датчиків, будівель, факультетів, поверхів, приміщень та публічної інформації з сайту.

Тепер приміщення можна створити з вибором кафедри або додати нові в панелі редагування.

2.3 Моделювання бази даних

База даних повинна мати наступні сутності:

- Будівлі, які мають поле з ім’ям, масив залежних поверхів, посилання на SVG файл з зображенням будівлі та адресу.
- Факультети, які мають поле з ім’ям, масивом залежних поверхів та масивом залежних кафедр.
- Поверхи, які мають номер, назву факультету, від якого вони залежать, масив з кімнатами, які знаходяться на поверсі, назва будівлі, колір факультету, посилання на логотип факультету.
- Кафедри, які мають назву, назву факультету та масив з залежними кімнатами.
- Кімнати, які мають номер, номер поверху, назву факультету, кількість місць, тип приміщення, масив з посиланнями на фотографії, опис, ім’я асистента, масив з пов’язаними кафедрами, масив з датчиками CO2, масив з датчиками температури, посилання на історію з датчиків CO2, посилання на історію з датчиків температури.
- Користувачі, які матимуть логін, пароль та роль.

2.4 Дизайн проекту

При обговоренні проекту було прийнято рішення про створення максимально схожого дизайну до попереднього сайту та взяття за основу деяких сторінок з того ж сайту. Кольорова схема буде використовуватись з попереднього сайту та з можливим використанням стилю головного сайту університету[28].

РОЗДІЛ 3
ТЕХНІЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Ініціалізація проекту, створення серверу.

В першу чергу було створено основну структуру серверу на NodeJs – основний файл серверу під назвою server.js та було ініціалізовано проект за допомогою терміналу та програмного засобу npm.

Також було встановлено базу даних MongoDB та користувацький інтерфейс MongoDBCompass.

Далі було встановлено основні бібліотеки для створення початкового серверу: express.js для створення шляхів та генерації сторінок, mongoose для роботи з базою даних MongoDB та path для зручної роботи з файлами.

Далі було імпортовано всі встановлені модулі та створено базовий код серверу, який дозволяв запускати поки що не взаємодіючий з користувачем сервер.

Наступним кроком було створення базового функціоналу CRUD по спроектованій раніше базі даних.
В першу чергу було створено файли моделей, має сенс розглянути реалізацію лише одної моделі, яка містить найбільшу кількість різних типів даних та параметрів, так як інші моделі матимуть схожий вигляд та інші моделі не матимуть відмінностей у реалізації, з одною лише відмінністю в назві та типі полів. Код представлений у лістингу 3.1.

```
const mongoose = require('mongoose');


const buildingSchema = new mongoose.Schema({
    name:{
        type: String,
        required: true,
        unique: true
```

```
    },
    floors:{
        type: [],
        required: true
    },
    svg:{
        type: String
    },
    address:{
        type: String
    }
});

module.exports = mongoose.model('Building', buildingSchema);
```

Лістинг 3.1. ./models/building.js

В першому рядку імпортується бібліотека mongoose, яка призначена для роботи з базами даних MongoDB. Далі ініціалізується схема(модель) поверху, де в аргументи передається JavaScript об’єкт, який містить необхідні поля у вигляді об’єктів, які в свою чергу містять параметри комірки в базі даних.

Першим елементом об’єкту є його назва, яка має текстовий тип даних, обов’язкова до заповнення та унікальна. Це необхідно для того, щоб уникнути дублікати та безіменні поверхи.

Далі йде комірка поверхів, які відносяться до будівлі. Створення відношення залежності буде реалізовано методом створення масиву з іменами залежних об’єктів, так як база даних MongoDB не розрахована на роботу з дочірніми елементами. Також комірка є обов’язковою, для

уникнення проблем зі створенням та заповненням масивів, при відсутності залежних від поверху сутностей, в даному випадку поверхів.

Наступним полем йде рядок з посиланням на svg файл з зображенням будівлі. Так як база даних не дозволяє напряму завантажувати в себе медіа об’єкти необхідно завантажувати їх окремо та додавати на них посилання.

Далі було реалізовано базовий функціонал CRUD для бази даних, використовуючи роути з бібліотеки Express.Js для створення зручного API.

Всі роути мають схожу реалізацію, тому в цьому випадку також є сенс розглянути лише один, який має найширший функціонал.

Переглянути повний код роуту можна в репозиторії проекту на GitHub[29].

В першу чергу було створено функцію отримання всіх об’єктів з бази даних у вигляді GET запиту. Його адреса виглядає наступним чином: назва-сайту/rooms, далі всі запроси матимуть схожий формат, окрім тих, де треба конкретизувати об’єкт, в такому випадку буде використовуватись з постфіксом /:number, де number це номер кімнати.

Лістинг 3.2 має код для GET запиту на отримання всіх кімнат та повернення файлу json у відповідь на запит.

```
router.get('/', async (req, res) => {
    try{
        const rooms = await Room.find();
        res.json(rooms);
    } catch (err){
        res.status().json({message: err.message});
    }
});
```

Лістинг 3.2. Роут Get для отримання всіх кімнат.

Цей запит буде в майбутньому використовуватись для формування списку всіх кімнат в панелі адміністрування сайтом.

Далі потрібно реалізувати можливість отримувати лише одну кімнату, цей потрібно для відображення конкретних сторінок та завантаження інформації лише про певну кімнату. Це буде використовуватись як в панелі адміністратора, для перегляду та редагування кімнати, так і для перегляду цієї кімнати користувачем.

В першу чергу було створено функцію пошуку приміщення в базі даних за її номером. Вона робить схожий до попереднього запит, але метод замінентй на findOne, для отримання лише одного об’єкту, а в аргументах передається об’єкт з полем number, якому присвоюється значення з отриманого в посиланні параметру /:number. Далі функція оброблює помилки і при успішності виконання привласнює значення цієї кімнати полю в об’єкті відповіді (response) і передає роботу наступному middleware.

Лістинг 3.3 містить реалізацію роуту отримання лише одного приміщення

```
router.get('/:number', getRoom, (req, res) => {
    res.json(res.room)
});
```

Лістинг 3.3. Роут отримання одної кімнати

Єдине що він робить, це викликає middleware з отриманням кімнати та відправляє користувачу відповідь з кімнатою у форматі JSON.

Наступним і одним з найважливіших роутів є роут створення нових приміщень. Він має тип POST та приймає два middleware, один з яких є

авторизацією користувача, а інший – системою для завантаження файлів. Систему авторизації буде розібрано після розбору та опису тестування всього функціоналу CRUD. Система завантаження файлів Multer буде описана в іншій частині роботи, на даному етапі інформація про цю систему буде надана в абстрактному вигляді.

Повний код цього роуту можна знайти в моєму репозиторії на GitHub[29], тут ми розберемо тільки деякі його частини.

В першу чергу було перевірено за допомогою методу класу схеми .exists(arg) на коректність введених факультетів та аудиторій і при наявності помилки записується значення false у змінні, та перевіряється у подальшому розгалуженні в коді.

Далі за допомогою циклу перевіряється масив з кафедрами і результат записується в аналогічну змінну.

Після цього ініціалізується масив з іменами завантажених зображень.

В наступній частині коду йде перевірка на коректність введених даних і у випадку негативного результату користувач отримуватиме JSON файл з помилкою та кодом помилки.

Далі формується об’єкт, який буде записаний в базу даних, його ми розглянемо більш детально, так як це важлива частина в реалізації продукту і аналогічний код буде використовуватись для інших видів даних в базі. Лістинг 3.4 містить код формування об’єкту.

```
const room = new Room({
        number: req.body.number,
        floor: req.body.floor,
        faculty: req.body.faculty,
```

```
            capacity: req.body.capacity,
            type: req.body.type,
            photo_links: images[0] != '' ? images : [],
            description: req.body.description,
            assistant: req.body.assistant,
            model: req.body.model,
            pulpits: req.body.pulpits[0] != '' ? req.body.pulpits : [],
            co2: req.body.co2[0] != '' ? h : [],
            temperature: req.body.temperature[0] != '' ? req.body.temperature : [],
            co2_history: req.body.co2_history[0] != '' ? req.body.co2_history : [],
            temperature_history: req.body.temperature_history[0] != '' ? req.body.temperature_history : [],
        });
```

Лістинг 3.4. Формування об’єкту кімнати перед відправкою в базу даних.

Він має аналогічні до описаних в розділі 2 поля. Кожному полю привласнюється аналогічне значення з тіла запита. В деяких випадках робиться додаткова перевірка для коректного збереження даних, через видозмінення їх методом передачі даних, сформованих не в JSON файл, а в FormData. Так, якщо масив з файлами опинився порожнім, то він матиме вигляд масиву з порожнім рядком, щоб це обробити треба зробити перевірку чи є перший елемент масива порожній рядок, якщо так, то він замінюється на порожній масив. Аналогічно з іншими масивами.

Далі йде частина з кодом, який зберігає вже сформовану кімнату в базі даних та редагує батьківські сутності додаючи в масив з посиланнями ім’я кімнати. Клієнту повертається json відповідь з бази даних та код 201, який інформує про успішне створення об’єкта.

Наступним йде роут оновлення існуючого об’єкта, який реалізований у вигляді запиту PATCH та приймає параметр /:number та виконує три middleware – автентифікація, завантаження файлу та пошук приміщення. Реалізація схожа з попереднім роутом. Головними відмінностями від попереднього роута є заміна файлів зображень на нові, та видалення старих. Дані до відправки формуються з отриманої з бази даних кімнати шляхом заміни полів на нові, якщо такі були надані. Повний код можна знайти у мене в репозиторії на GitHub[29].

Останнім роутом функціналу CRUD є видалення приміщення з бази даних. Ознайомитись з кодом можна також в моєму репозиторії GitHub[29].
Запит має тип DELETE, приймає параметр з номером кімнати, викликає два middleware з авторизацією користувача та отриманням кімнати.

В першу чергу номер кімнати видаляється з масивів батьківських елементів, потім видаляються всі пов’язані з приміщенням файли а потім викликається метод схеми .remove(), а користувачу повертається повідомлення про успішне або не успішне видалення кімнати.

3.2 Тестування CRUD функціоналу за допомогою утіліти Postman

Утиліта Postman призначена для створення різних видів запитів та відображення отриманої інформації, в основному використовується як спосіб тестування API під час розробки програмних засобів.

В тексті буде розглянутий принцип тестування лише одного типу даних, так як функціонал інших має аналогічний вигляд.

В першу чергу треба перевірити функцію отримання всіх об’єктів типу «Room», для цього потрібно створити Post запит за створеним посиланням.

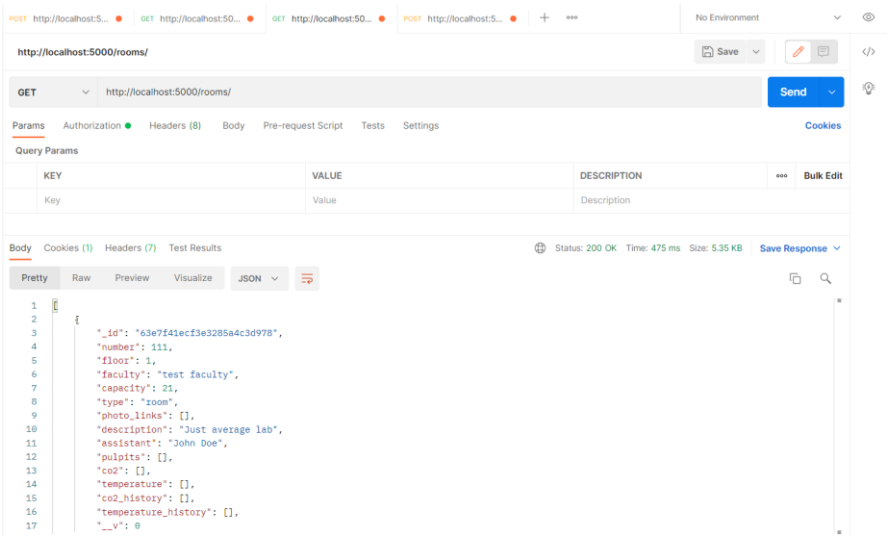

Рис. 3.1 Інтерфейс Postman, отримано результат за запитом.

В результаті було успішно отримано масив зі створеними в інтерфейсі MongoDB об’єктами.

Далі необхідно перевірити роботу GET запит на отримання лише одного елементу за його назвою. Для цього потрібно додати до існуючого посилання символ “/” з номером приміщення.

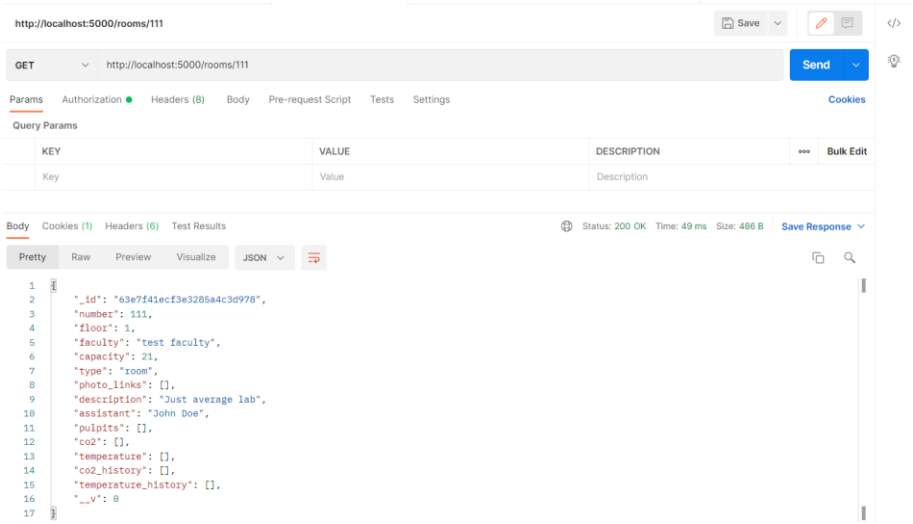

Рис. 3.2 Результат запиту на отримання одного об’єкту

Цей запит також коректно повертає користувачу об’єкт.

Далі було протестовано запит створення нового об’єкту в базі даних. Був відправлений запит POST, який мав тіло аналогічної будови до визначеної в розділі 2 структури. В базі даних було створено необхідий об’єкт

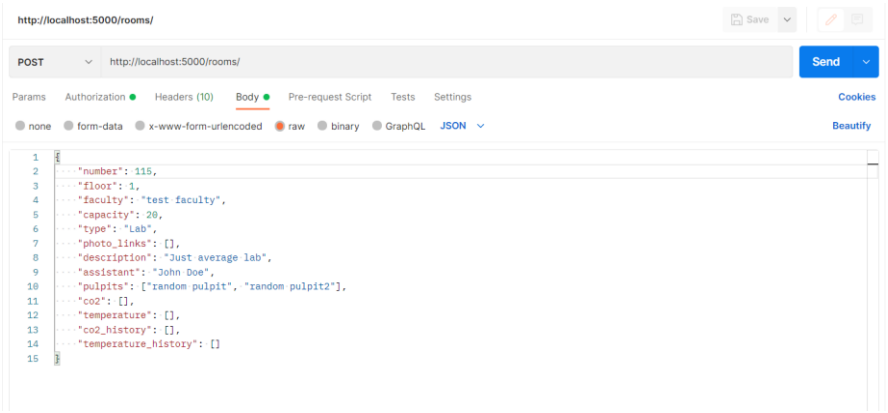

Рис. 3.3 Запит створення нового об’єкту у базі даних

Запит видалення має аналогічний вигляд до запиту отримання одного об’єкта за виключенням іншого позначення запиту DELETE.

Після виконання запиту приміщення було успішно видалено, об’єкт зник в інтерфейсі MongoDBCompass і користувач отримав відповідь у вигляді JSON файлу, який сигналізував про успішність операції.

Метод редагування мав аналогічний до методу створення вигляд, але в посилання було додано символ “/” та номер кімнати, а типом запиту було встановлено «POST». Після редагування об’єкт було змінено, що відображалось в інтерфейсі MongoDBCompass. Користувач отримав відповідь про успішне виконання операції.

3.3 Створення системи авторизації

Для створення системи авторізації необхідно створити базу даних користувачів та тестового користувача. На даний момент, для швидкої

реалізації інших функцій проекту було створено версію без різних рівнів доступу, але в майбутньому такий функціонал буде імплементовано.

Модель користувача в mongoose має такий вигляд, як код на лістингу 3.5

```
const mongoose = require('mongoose');
const User = new mongoose.Schema({
    login: {type: String, unique: true, required: true},
    password: {type: String, required: true}
})
module.exports = mongoose.model('User', User);
```

Лістинг 3.5. Код схеми користувача

Далі, по аналогії з іншими об’єктами в базі даних було частково створено CRUD функціонал. Запит створення профілю має посилання у вигляді Базовий URL сайту + “/auth/register” та перевіряє ім’я користувача на існування. Далі дані користувача шифруються за допомогою бібліотеки bcryptjs з параметром salt: 7. Зашифрований пароль та логін зберігаються у базу даних.

Далі було створено також POST запит, але вже на авторизацію користувача, він має структуру у вигляді Базовий URL сайту + “/auth/login”. Та приймає два поля в тілі запиту – login та password. Далі проводиться пошук по логіну користувача і при наявності такого код переходить на наступний етап – порівняння паролей за допомогою метода compareSync бібліотеки bcryptjs. При верифікації всіх даних генерується токен користувача та повертається користувачеві. Цей токен буде використовуватись для безперервної роботи користувача у певних часових рамках. Токен записується в cookie файли користувача.

Першого користувача було створено за допомогою утіліти Postman, він має тестові логін та пароль та його присутність необхідна лише в цілях розробки основного функціоналу. Після було протестовано систему логіну та користувач успішно пройшов авторизацію.

Далі необхідно створити систему автентифікації користувача. Її можна реалізувати у вигляді middleware. Лістинг коду можна знайти на GitHub сторінці цього проекту[30].

Робота цього middleware виглядає наступним чином:

Перевіряються кукі файли користувача, якщо поле token не існує, то користувач перенаправляється на сторінку з вводом пароля. В іншому випадку токен перевіряється бібліотекою jsonwebtoken і обирає користувача, від чийого імені буде проводиться перегляд сторінки. В непередбачуваних випадках користувач буде перенаправлятись на сторінку входу.

3.4 Створення дизайну для панелі адміністрування системою

Далі необхідно спланувати розробку фронтенд частини сайту та створити дизайн, це необхідно для зручного користування, а створення шаблонів сторінок пришвидшить розробку системи.

Для створення дизайну було використано безкоштовну версію програмного забезпечення Figma[31].

При розробці дизайну було прийнято рішення орієнтуватись на стиль університету, головний сайт університету, попередню версію сайту, побажання керівництва університету та отриманий досвід від аналізу сайтів в першому розділі роботи.

В першу чергу було розроблено палітру кольорів для майбутнього дизайну. Світлий синій колір було обрано, так як він вже був на попередньому сайті, а побажанням керівництва було збереження стилю

попереднього сайту якомога сильніше. Світло сірий колір також було обрано на основі кольору фону сайту.

Палітра мала вигляд, зображений на Рис. 3.4

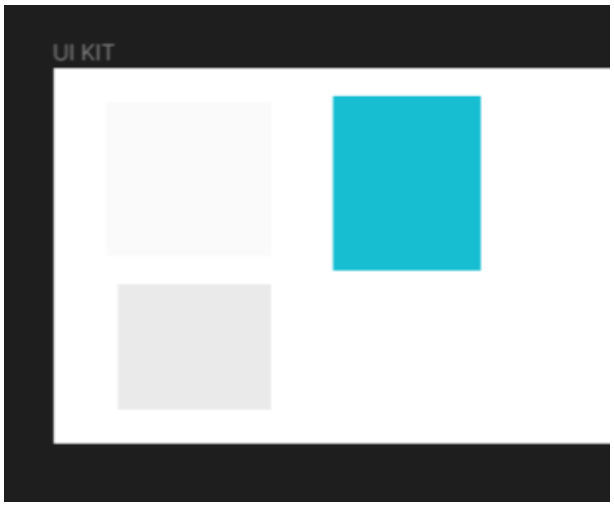

Рис. 3.4 Палітра сайту

Далі було розроблено панель адміністратора у вигляді дешборду, з урахуванням тенденцій ринку. Порожня сторінка- шаблон складається з меню, яке було скопійовано з дизайну минулого сайту, бокового меню зі сторінками в панелі адміністратора та основної частини, де буде розміщуватись інформація та поля для вводу даних.

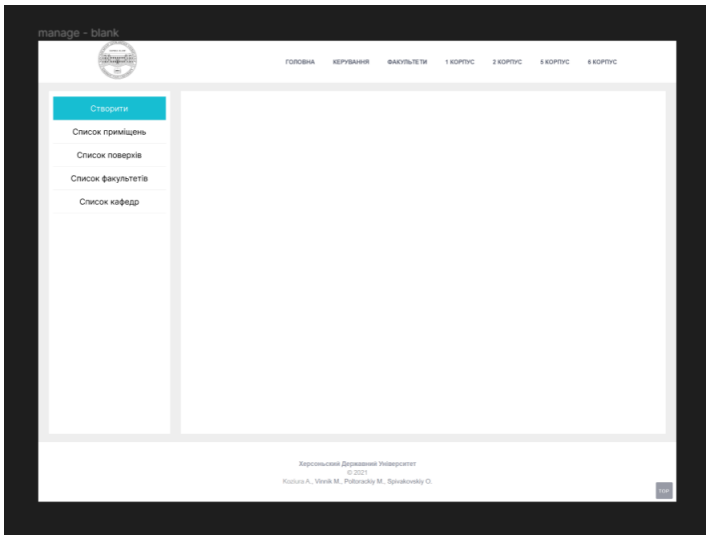

Рис. 3.5 Сторінка-шаблон дешборду

Далі було створено дизайн сторінок створення нового об’єкту та редагування старого на основі вже створеного шаблону. Вони мають майже ідентичний вигляд, за винятком різного набору полів.

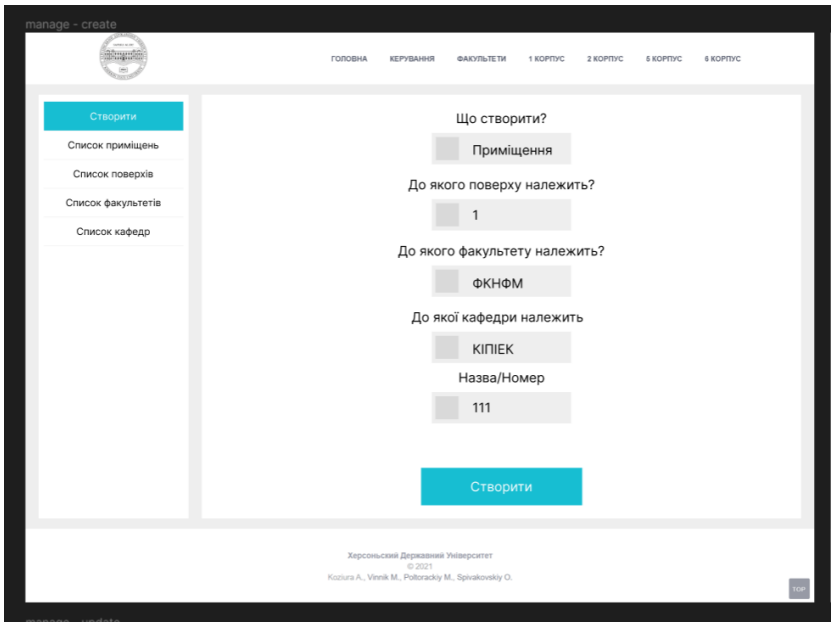

Рис. 3.6 Сторінка створення нового об’єкту.

Далі було створено дизайн сторінки зі списком об’єктів, він однаковий для всіх за винятком різних полів, які будуть виводитись.

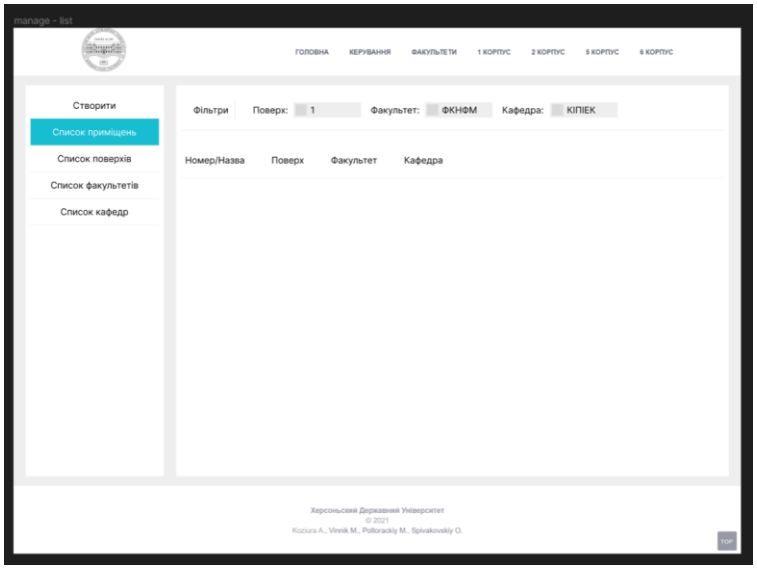

Рис. 3.7 Дизайн сторінки зі списком об’єктів

3.5 Імпортування фронтенд частини попереднього сайту

Наступним етапом буде завантаження попередньої версії сайту. Всі файли було збережено в окрему папку зі збереженням структури сайту.

Далі цей проект було інтегровано в Gulp збірку для зручної верстки. Збірка включала в себе такі технології як препроцесор SCSS[32] та сервер, який відображає зміни сторінки в реальному часі[33].

Далі було зверстано три сторінки за створеним раніше дизайном: сторінка створення, сторінка списку, сторінка редагування. Також було відредаговано код навігаційного меню та футеру сайту, для відображення пункту «Керування» та було внесено декілька косметичних змін, таких як вирівнювання логотипу по центру, створення безшовної анімації зміни кольору логотипу та видалення мітки про поточну сторінку на сайті. Також було зверстано сторінку входу на сайт без створення дизайну, вона має два поля та кнопку підтвердження даних.

На даний момент сайт має готовий фронтенд для реалізації основних функцій сайту, тому створений за допомогою Gulp результат переноситься в папку static проекту, а сторінки панелі адміністратора були перенесені в окрему папку views та переведені у формат ejs, для роботи з динамічними сторінками.

3.6 Створення сторінки входу для користувача

В першу чергу створюється вже працююча сторінка входу на сайт. Для цього її треба перевести до формату ejs та створити необхідний роут. Сторінка матиме 2 поля, кнопку активації та скрипт відправки даних користувача на сервер. Сторінка вхожу знаходитиметься за адресою Базовий URL сайту + “/login”. Роут лише надає сторінку з логіном користувачеві. Сторінка реалізована з використанням фреймворку VueJs. Код скрипта логіну знаходиться на моєму GitHub репозиторії в папці зі статикою[34]. При заповненні полів формується об’єкт користувача, який

відправляється за POST запитом входу на сайт. При отриманні позитивного результату в cookie файли записується токен для роботи та скрипт перенаправляє користувача на сторінку зі створення нових об’єктів.

Тепер при відкритті будь якої сторінки, яка використовує auth middleware користувач перевіряється на те чи увійшов він на сайт і при негативному результаті він направляється на сторінку входу, яка виглядає так, як показано на Рис. 3.8

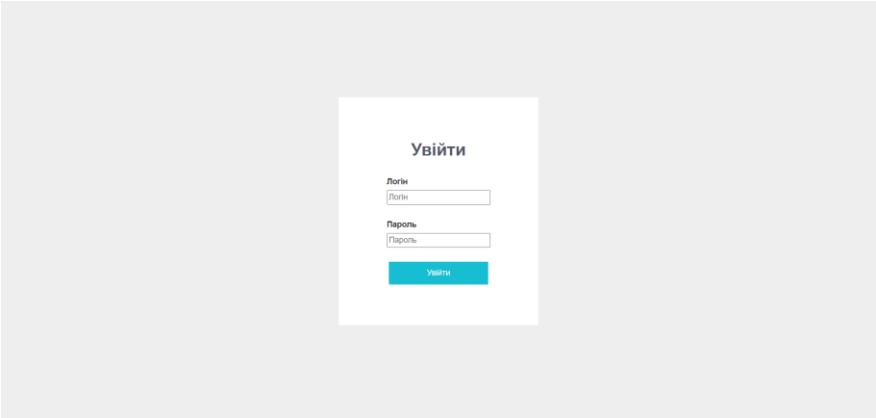

Рис. 3.8 Сторінка входу на сайт

3.7 Створення сторінок зі списками об’єктів в базі даних

Далі було створено п’ять сторінок, які отримують інформацію з бази даних. При заході на сторінку JavaScript код з фреймворком VueJs робить звернення до серверу з запитом отримати всі об’єкти необхідного типу.

Код отримання всіх кімнат з бази даних знаходиться у лістингу 3.6

```
const App = {
    data(){
        return{
            items: []
        }
    },
    async mounted(){
```

```
        fetch('/rooms/', {
            method: 'GET',
            headers:{
                'Content-Type': 'application/json'
            },
        }).then(res =>{
            return res.json();
        }).then(data=>{
            this.items = data;
        });
    }
}

Vue.createApp(App).mount('#dash-body');
```

Лістинг коду 3.6 Скрипт отримання кімнат з серверу

Потім отримані дані виводяться на сторінку за допомогою циклу VueJs та підстановкою даних за допомогою Vue.

Також кожен елемент оформлений у вигляді посилань, які генеруються використовуючи базове посилання сайту та номер кімнати.

Створена сторінка відображена на Рис. 3.9

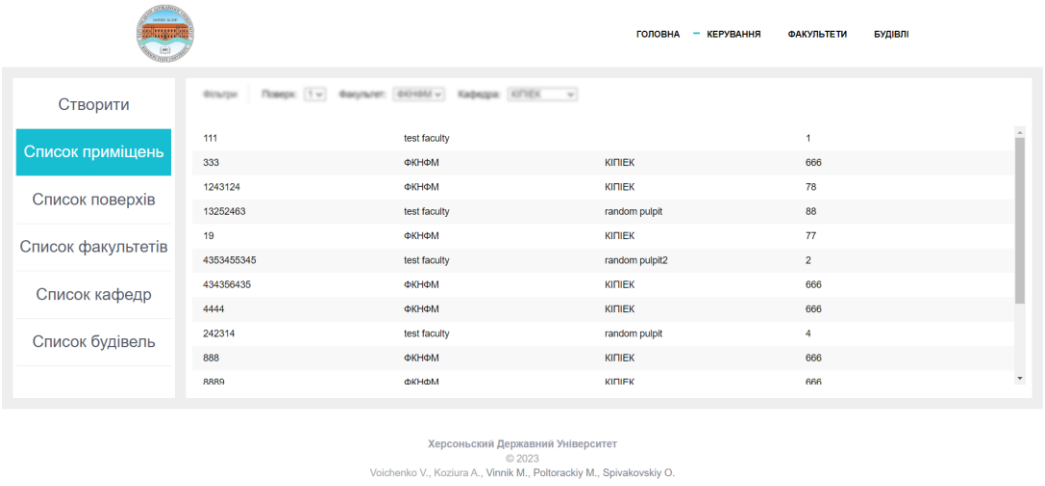

Рис. 3.9 Сторінка зі списком кімнат

3.8 Створення динамічного меню дешборду

Для того, щоб користувач міг зручно орієнтуватись в структурі сайту треба зробити динамічне меню, яке буде вказувати користувачу на його позицію на сайті. Для цього треба передавати інформацію про те, до якої категорії відноситься поточна сторінка. Цю інформацію можна передавати через EJS. Реалізувати динамічне меню можна також за допомогою EJS розбиваючи сторінку на елементи Partials.

Для цього потрібно створити файл ejs в каталозі partials, який буде знаходитись в каталозі views.

Цей файл міститиме в собі HTML код меню та код підстановки значень з EJS. Переглянути код можна в GitHub репозиторії проекту[26].

На Рис. 3.10 зображене підсвічення сторінки поверхів, при знаходженні на сторінці поверхів.

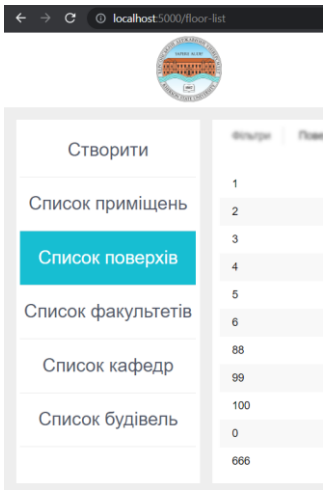

Рис. 3.10 Бокове меню дешборду

3.9 Створення сторінки додавання нових об’єктів

Наступною буде створена сторінка створення нових елементів в базу даних. Користувача буде зустрічати випадаючий список з типом об’єкту, який користувач бажає створити.

Переглянути код можна в файлі manage.js в папці static/js/ на GitHub сторінці проекту.

Скрипт на VueJs буде перевіряти вибір користувача та увімкне ті поля, які потрібні для створення саме цього елементу.

При виборі факультету у користувача змінюється доступний набір поверхів та кафедр, які належать до цього факультету.

Всі поля у фронтенді пов’язані з моделлю VueJs і оновлюються одночасно.

При заповненні всіх необхідних полів користувач має змогу підтвердити введену інформацію натисканням кнопки, яка викликає функцію відправки даних на сервер. В залежності від обраного типу об’єкту скрипт обирає алгоритм та адресу, на яку буде відправлено дані з полів.

Після відправки даних користувач отримує або помилку або підтвердження успішності операції.

Об’єкти, які потребують завантаження зображень відправляються у вигляді FormData, а не у форматі JSON. Такий тип дозволяє передавати зображення до серверу, але має певні нюанси у обробці даних, такі як виникнення порожніх рядків у масивах.

При додаванні файлів викликається функція обробки цих файлів, яка привласнює поле files у змінну js коду.

Далі сформований JSON файл або FormData відправляються на сервер, після чого користувач отримає повідомлення про успішність або неуспішність дії.

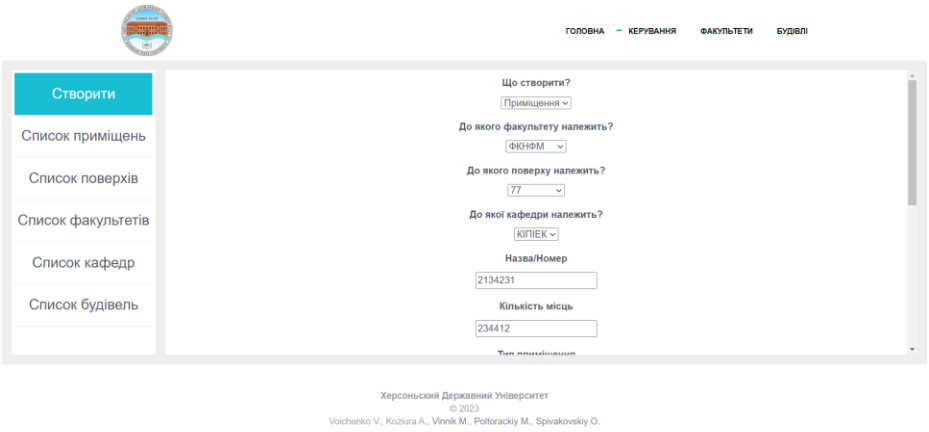

Рис. 3.11 Приклад заповненої сторінки

3.10 Створення сторінок перегляду об’єктів

Реалізація цих сторінок буде базуватись на запиті отримання одного об’єкту по його назві. Поля об’єкту будуть передаватись через EJS і інтегруватись в сторінку за допомогою нього ж. Номер або назва сторінки буде братись з адреси сторінки.

При створенні html елементу буде проводитись перевірка на існування поля і у випадку його відсутності заголовок не буде відображатись користувачу.

Всі сторінки мають аналогічну реалізацію окрім видів полів, які будуть залежать від типу об’єкту, який користувач намагається подивитись.

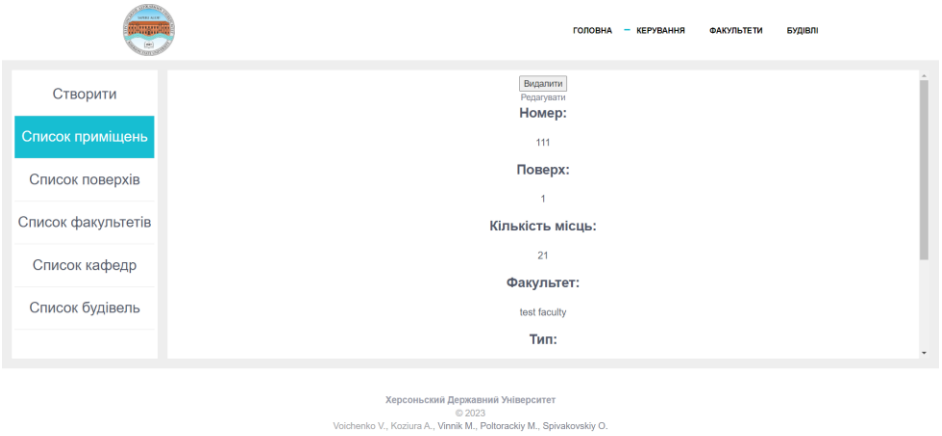

Рис. 3.12 Сторінка перегляду кімнати

3.11 Створення функціоналу видалення та об’єктів на сторінках перегляду

На всі сторінки перегляду об’єктів було додано кнопку, при натисканні на яку викликалась функція з запитом DELETE і назвою цього об’єкту. Сайт запропонує користувачу підтвердити свій вибір і при позитивному результаті запит на видалення буде надіслано. В результаті користувач отримає повідомлення про успішність або неуспішність операції.

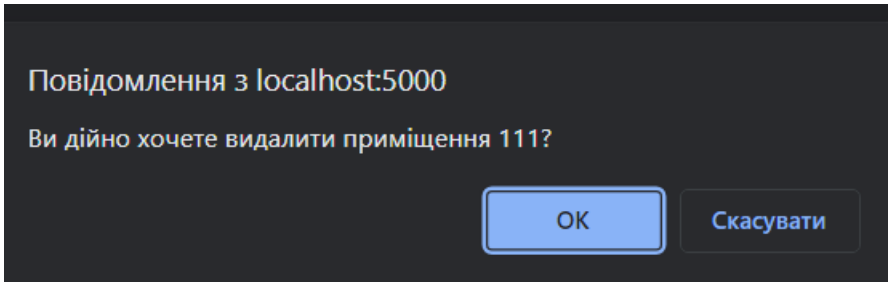

Рис. 3.13 Підтвердження видалення приміщення

3.12 Створення сторінки редагування приміщень

Сторінка редагування приміщень була розроблена на основі коду сторінки створення об’єктів, за виключенням видалення поля з вибором типу об’єкту та деяких полів, які не повинні редагуватись. При завантаженні зображень старі видаляються та заміщуються.

Потрапити на сторінку редагування можна за посиланням на сторінці об’єкту.

При заході на сторінку старі дані автоматично додаються в усі поля.

При натисканні на кнопку «Зберегти» скрипт відправляє оновлені дані на сервер методом PATCH. Після натискання користувач отримає сповіщення про успішність операції.

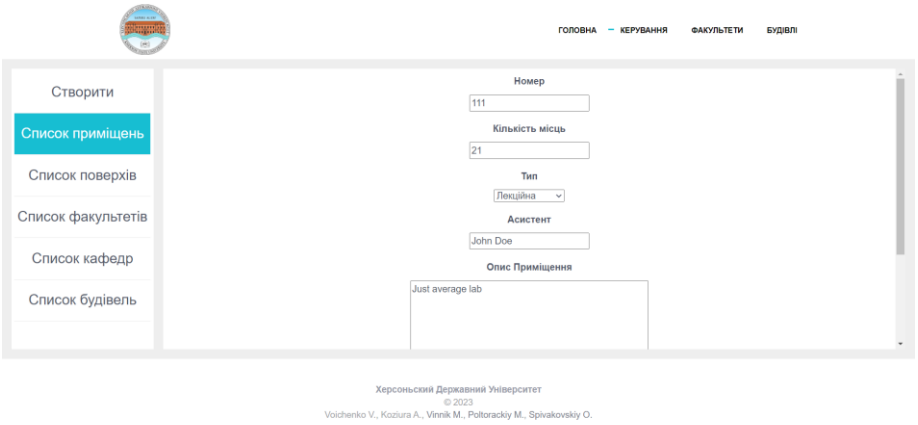

Рис. 3.14 Сторінка редагування приміщення

3.13 Створення системи завантаження файлів

Система завантаження файлів базується на бібліотеці Multer, яка дозволяє зручно створювати такі системи і інтегрувати їх в будь яку базу даних.

Для цього створюється middleware з завантаженням файлів, в якому ініціалізується multer та обираються шляхи, де будуть зберігатись файли. Код ініціалізації присутній в Лістингу 3.7

```
const storage = multer.diskStorage({
    destination: (req, file, cb) =>{
        cb(null, './static/images');
    },
    filename: (req,file,cb)=>{
        cb(null, Date.now() + path.extname(file.originalname));
    }
});

const upload = multer({
  storage: storage
});
```

Лістинг 3.7 Ініціалізація Multer

3.14 Перетворення сторінок сайту на EJS файли, фронтенду сторінок

Наступним йде етап підготовки до розроблення нової системи сторінок основної частини сайту для користувача. По перше вони переводяться в формат EJS як і сторінки панелі адміністрування.

Далі створюються порожні файли меню та футеру сайту, які в майбутньому стануть основою для динамічних футеру та меню. Код існуючого футеру та меню переноситься у відповідні файли. Прибираються зайві коментарі та рядки коду.

3.15 Створення динамічного навігаційного меню сайту та футера

Аналогічно до бокового меню дешборду на кожну існуючу сторінку передається інформація про залежність її до певної категорії та за допомогою умовного оператору VueJs додається клас активної сторінки необхідному елементу меню на сайті. На Рис. 3.15 зображено виділення категорії «Будівлі» при знаходженні на сторінці будівель.

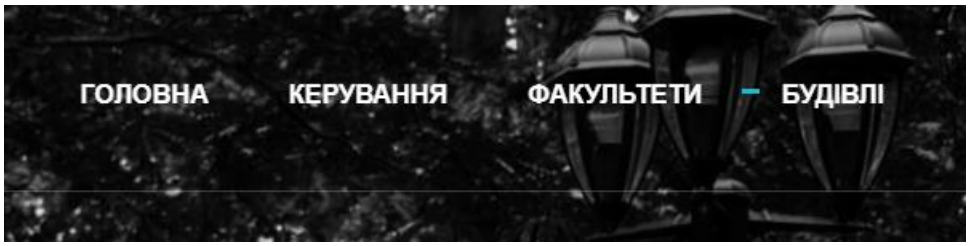

Рис. 3.15 Меню сайту

Динамічний футер реалізовано аналогічним шляхом, але без використання інформації з серверу. При оновленні одного файлу футеру будуть змінюватись абсолютно всі футери на сайті.

Рис. 3.16 Динамічний футер

3.16 Створення нової системи сторінок

Всі сторінки необхідно перевести зі статичних у динамічні зі створенням окремих роутів та з використанням інформації з серверу. Код роутів та сторінок можна знайти на GitHub сторінці проекту[26].

Сторінки будівель, поверхів та кімнат тепер генеруються схожим до панелі адміністрування способом. На сторінці «Будівлі» генерується список з посиланнями на сторінки будівель.

Рис. 3.17 Згенерований список будівель університету

Сторінки будівель генеруються таким же чином, як і сторінки з переглядом будівель в панелі адміністратора. Зліва відображається SVG файл, який завантажується користувачем, а справа список поверхів з позначенням факультету. При натисканні на елемент списку користувача направляє на сторінку поверху.

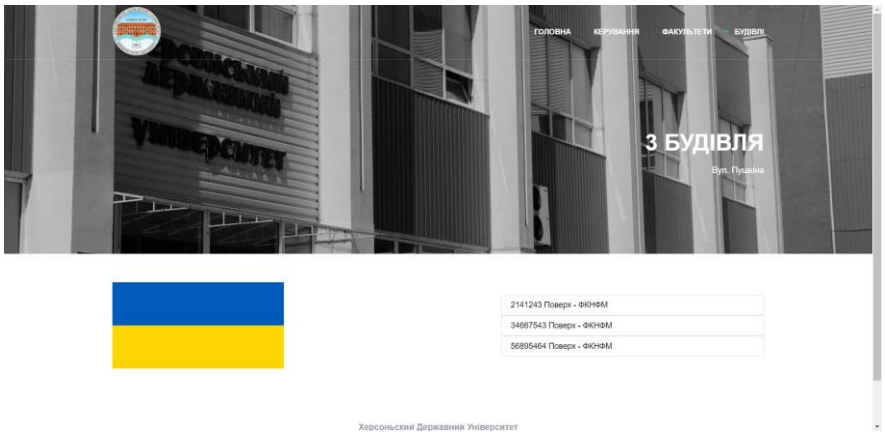

Рис. 3.18 Сторінка будівлі з SVG зображенням-плейсхолдером

3.17 Створення сторінки поверху

За основу нової сторінки поверху була взята сторінка з попереднього сайту, на яку були додані поля, які заповнюються динамічно при завантаженні сторінки. Колір фону береться з бази даних.

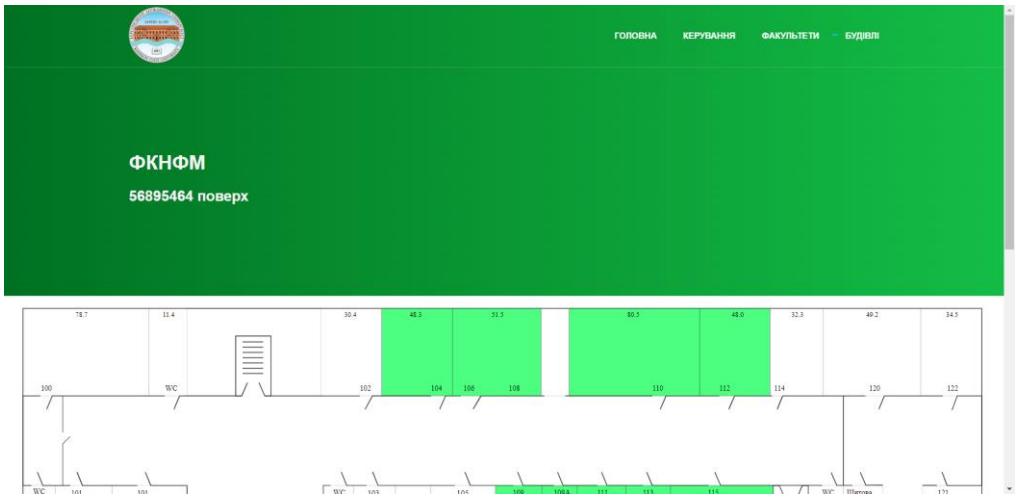

Рис. 3.19 Сторінка поверху.

На цій сторінці є перспектива створення списку аудиторій, які залежать від цього поверху, або динамічне додавання посилань в SVG файл поверху.

3.18 Створення сторінки приміщення

Сторінка з приміщеннями реалізована аналогічним способом до подібної сторінки в палені адмініструванням. За основу був взята сторінка поверху та перероблена під інформацію з кімнати.

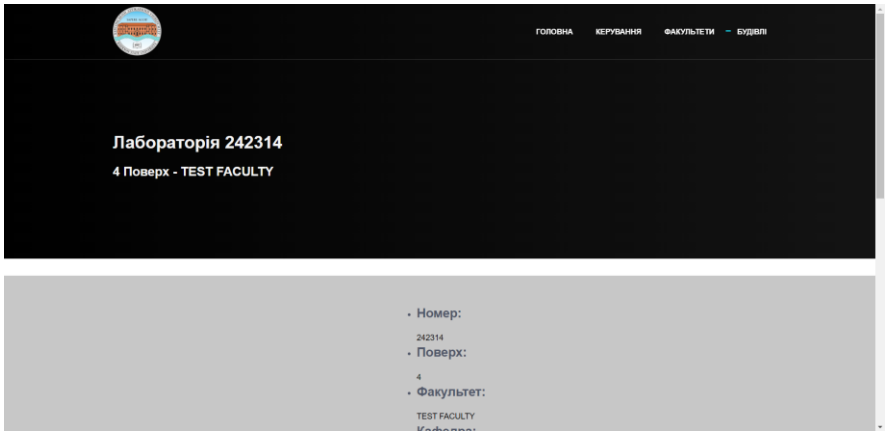

Рис. 3.20 Сторінка приміщення без зображення

3.19 Система моніторингу збитків

Також було створено систему моніторингу збитків, яка має аналогічну з системою створенням, переглядом та редагуванням об’єктів реалізацію. Ця система має перспективи розвитку у автоматичному рахуванні збитків по категоріям та створенню звітів.

3.20 Створення системи різних ролей користувачів

Проект має перспективу розвитку у створенні різних рівнів доступів для користувачів. Так, лаборант може редагувати аудиторії, де він призначений, а різні групи працівників матимуть можливість додавати та редагувати лише у дозволених факультетах та поверхах.

3.21 Система сповіщень про перетин порогових значень

Система має перспективу розвитку системи сповіщень персоналу, що дозволить оперативно реагувати на проблеми в приміщеннях та направляти персонал в проблемні місця.

ВИСНОВКИ

В роботі було проаналізовано популярні релевантні рішення в сфері інтернету речей. На основі проаналізованих сервісів було розроблено SWOT аналіз та виділено функції, які варто було б додати до проекту. Також було визначено напрямок у створенні дизайну та зручності для користувача. Ні один з проектів не вирішував всі питання, тому було прийнято рішення створити вимоги на основі різних проектів, з яких було виділено найважливіший функціонал.

В даному проекті було створено:

- Базу даних на основі MongoDB
- Базовий сервер на NodeJS
- Створено CRUD функціонал для роботи з базою даних
- API для бази даних.
- Систему користувачів та авторизації
- Систему різних рівнів доступу до даних
- Дизайн сторінок панелі адміністратора в Figma
- Сторінки панелі адміністратора
    - Сторінка створення нових об’єктів
    - Сторінки зі списками всіх наявних видів об’єктів у базі даних
    - Сторінки перегляду об’єктів
    - Сторінка редагування приміщення
- Сторінку входу для користувача
- Нові файли фронтенду на основі попереднього сайту
- Динамічні меню навігації, меню дешборду та футер
- Систему завантаження файлів
- Сторінку приміщень для гостей
- Нову систему сторінок з будівлями, поверхами та приміщеннями

- Систему моніторингу збитків.
- Систему сповіщень адміністратора

Проект має перспективи розвитку у вигляді збільшення різновиду датчиків, сторінок з моніторингу та менеджменту інших сфер університету, таких як місячні витрати, керування точками WiFi, можливістю віддалено керувати світлом та розетками, отримувати інформацію з лічильників та моніторинг котельні. На цьому система може не припинити розвиток і розширитись до мобільного додатку для студентів, який допоможе знайти аудиторію, дізнатись про заповненість коворкінгу тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://iot-analytics.com/number-connected-iot-devices/
2. https://www.kerlink.com/blog/2022/03/09/utilities-technology-trends-2022-iot-enables-improved-operational-performance/
3. https://chat.openai.com/chat
4. Mi Home https://play.google.com/store/apps/details?id=com.xiaomi.smarthome&hl=uk&gl=US
5. CitiMan by Dhyan Networks and Technologies https://www.dhyan.com/solutions/smart-city-central-management-system/
6. Microsoft Azure https://azure.microsoft.com/
7. ThingsBoard website https://thingsboard.io/
8. ThingsBoard Cloud https://thingsboard.cloud/
9. Smart Campus Sejong GitHub repository https://github.com/seslab-sju/Smart-Campus-Sejong-University
10. Дослідження можливостей розробки Smart Campus від Sònia Gudayol Marquès https://upcommons.upc.edu/bitstream/handle/2117/89403/TFG_SoniaGudayol.pdf
11. Войченко В. В. Проектування Сервісу «Smart University» : курсова робота / В. В. Войченко ; наук. керівник к.пед.н. доц. М.О. Вінник ; Міністерство освіти і науки України ; Херсонський держ. ун-т, ф-т комп'ютерних наук, фізики та математики, кафедра комп'ютерних наук та програмної інженерії. – Херсон : ХДУ, 2022. – 27 с.
12. OpenWeather API https://openweathermap.org/api
13. Mobius app GitHub repository https://github.com/IoTKETI/Mobius
14. Документація до NodeJs https://nodejs.org/en/docs
15. Документація до ExpressJs https://expressjs.com/

16. Документація Mongoose https://mongoosejs.com/docs/api/mongoose.html
17. Документація Multer https://www.npmjs.com/package/multer
18. Сторінка cookie-parser https://www.npmjs.com/package/cookie-parser
19. Сторінка bcrypt https://www.npmjs.com/package/bcrypt
20. Сторінка jsonwebtoken https://www.npmjs.com/package/jsonwebtoken
21. Сайт та документація до VueJs https://vuejs.org/
22. Сайт Bootstrap https://getbootstrap.com/
23. Сайт та документація до Gulp https://gulpjs.com/
24. Сайт та документація до MongoDB https://www.mongodb.com/docs/
25. Документація до Postman https://learning.postman.com/docs/introduction/overview/
26. GitHub репозиторій проекту https://github.com/FalloutEngineer/smart-university
27. Документація до Ngrok https://ngrok.com/docs
28. Сайт Херсонського Державного Університету https://www.kspu.edu/default.aspx
29. Роут роботи з приміщеннями https://github.com/FalloutEngineer/smart-university/blob/master/routes/rooms.js
30. Middleware автентифікації https://github.com/FalloutEngineer/smart-university/blob/master/routes/auth.js
31. Сторінка з документацією до Figma https://help.figma.com/hc/en-us
32. Документація до SCSS https://sass-lang.com/documentation/
33. Документація до gulp-live-server https://www.npmjs.com/package/gulp-live-server
34. Код сторінки входу https://github.com/FalloutEngineer/smart-university/blob/master/static/js/login.js
35. Acuña, Leonardo & Narváez, Ray & Salas, Carlos & Magre, Luz & González, María. (2021). Smart UTB: An IoT Platform for Smart Campus. 10.1007/978-3-030-86702-7_21.

https://www.researchgate.net/publication/354916569_Smart_UTB_An_IoT_Platform_for_Smart_Campus

ДОДАТКИ

Додаток 1. Кодекс Академічної Доброчесності

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Войченко Владислав Володимирович,

учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

· вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;

· принципів та правил академічної доброчесності;

· нульової толерантності до академічного плагіату;

· моральних норм та правил етичної поведінки;

· толерантного ставлення до інших;

· дотримуватися високого рівня культури спілкування;

– надавати згоду на:

· безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;

· оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;

· використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науководослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

12.09. 2019		Владислав Войченко
(дата)	(підпис)	(ім'я, прізвище)